Operation/Service Manual

data systems

HEATH
Schlumberger

TRLR

HEATH
Schlumberger

Model WH8
DIGITAL COMPUTER

595-2149-01

CONTROLLED
DOCUMENT

SHOP
COPY

HEATH COMPANY
BENTON HARBOR, MICHIGAN 49022
a Schlumberger company

Copyright © 1978
Heath Company
All Rights Reserved
Printed in the United States of America

TABLE OF CONTENTS

INTRODUCTION 3

TEST ROUTINES
Initial Test Routine 4
Memory Test Routine 9

TROUBLESHOOTING
Precautions for Troubleshooting 15
Troubleshooting Charts 17

OPERATION
Introduction 25
Modes of Operation 26
  Decimal Point Operation 26
  Split Octal Display 27
Use of the Front Panel 28
  Register Mode 28
  Memory Mode 28
  Cancel 30
  Loading and Dumping Data 30
  Executing a Saved Program 31
  Inputting and Outputting 31

THEORY OF OPERATION
System Description 32
CPU Circuit 33
Front Panel (Control Circuit) 36
Power Supply 38
Instruction Set 39

SYSTEM CONSIDERATIONS
  Memory Map 56
  I/O Port Map 57
  Bus Functional Pin Definitions 57
System Configurations 58

APPENDIX
Source Program for the Memory Test Routine .. 61
The Functions of a Computer 63
The 8080 Central Processor Unit 68

Specifications 90

Semiconductor Component Number Index 91

Semiconductor Part Number Index 92

PARTS LIST 99

CIRCUIT BOARD X-RAY
VIEWS ..... (Illustration Booklet Page 5)

SCHEMATICS ..... Fold-in

WARRANTY ..... Inside rear cover

SERVICE INFORMATION ..... Inside rear cover

INTRODUCTION

NOTE: Before you proceed, you must have at least one memory circuit board installed in your Computer.

Your H8 Digital Computer is an 8-bit microcomputer that uses the popular 8080A microprocessor. The complete, low-cost, hobbyist Computer consists of an assembled and tested unit, exclusive front panel, and a power supply capable of supporting the H8 and many accessories.

The front panel includes a 16-key keyboard, four status lamps, and a 9-digit octal display for reading and altering memory and register contents. Also included in the front panel design is a 1K ROM monitor.

The CPU features 7-level vector interrupt capability, the standard 8080 instruction set, a fully buffered bus requiring no additional drivers, and 64K of memory addressing.

The monitor features automatic memory sizing and input/output initialization upon power-up, and load and dump routines which eliminate the need for boot strap and loader programs. Single instruction operation is also featured for testing and debugging programs. The monitor remains active during program execution to continuously monitor the status of the registers and memory.

Your Computer requires some additional memory before it is capable of operating. If you wish to communicate with the Computer through a terminal device, you must install a serial or parallel I/O interface. If no terminals are desired, programs can be entered and executed through the front panel keyboard.

The Computer uses a 50-line bus oriented design with ten locations which can be used for memory, parallel I/O interface, serial I/O interface, and other options.

OPERATION NOTES:

NOTE: Refer to the individual circuit board Manuals for detailed circuit board installation.

1. Do not remove or install circuit boards or components with the power on.

2. Always position the CPU circuit board in the P2 location.

3. Do not install circuit boards in the P10 location. This location is for expansion only.

4. Locate I/O circuit boards from the P9 location toward the front.

5. Locate memory circuit boards from the P3 location toward the back.

6. Locate circuit boards in alternate positions for improved ventilation. As more circuit boards are added, the power supply voltage will decrease, thus decreasing the dissipation in the circuit board regulators. Therefore, the remaining locations can be used without overheating.

7. Do not restrict ventilation. The H8 is convection cooled. Therefore the air vents, top and bottom, should not be obstructed.

8. Keep the low/normal switch in the "NORM" position until you are positive the line voltage is low.

TEST ROUTINES

Your Computer was carefully assembled and checked at the factory prior to shipment. If you wish to verify that the Computer was not damaged during shipment, you may wish to perform the following "Test Routines." If you do not wish to perform the following checks, proceed to "Operation."

It is not necessary, at this time, to have a working knowledge of your Computer. If, at any time during the "Test Routines," you fail to obtain the proper results, refer to the "Troubleshooting" section.

INITIAL TEST ROUTINE

This routine performs an initial check on your H8 Computer. A series of nine character messages will be displayed on the front panel LED's. The number of messages and the delay between them is variable.

The routine is entered in machine language through the front panel starting at address 40100. Remember to always enter a 6-digit address. To enter address 40100, you must enter 040100.

The following chart will help you begin to enter the "Initial Test Routine." You will be given detailed instructions and examples of each step as the H8 is turned on; the memory mode is entered; and the location is addressed, altered, and checked.

Press the keys in the "Keys Pressed/Result" column in a sequence from left to right as shown. The "Display" column shows the display you will observe on the front panel LED's.

NOTE: In the following chart, X=random number.

Refer to Pictorials 6-1 and 6-2 (Illustration Booklet, Pages 1 and 2) for the location of the front and rear panel features.

KEYS PRESSED/RESULT	DISPLAY
Power switch ON (on rear panel). A medium beep; random display.	XXX XXX XXX
MEM # RTM/Ø A short beep; all decimal points light.	X.X.X. X.X.X. X.X.X.
Ø DE 4 GO Ø SP 1 Ø Ø A short beep as you enter each digit and a medium beep as each 3-digit octal number is completed. When you enter the sixth digit, the decimal points go out.	040 100 XXX

KEYS PRESSED/RESULT	DISPLAY
ALTER / RST / Ø — A short beep; the decimal points scan from right to left.	040 100 XXX
Ø 7 SI PC 6 OUT — A short beep as you enter each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location.	040 101 XXX
− — A short beep; the memory address decrements one location; the contents of address 040 100 are displayed in the DATA/REGISTER LED's.	040 100 076
+ — A short beep; the memory address increments one location.	040 101 XXX
Ø Ø AF 2 — A short beep as you enter each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location.	040 102 XXX
− — A short beep; the memory address decrements one location; the contents of address 040 101 are displayed in the DATA/REGISTER LED's. NOTE: You can alter the contents of the memory location if an error has been made. For example: If you accidentally entered 003 at location 040 101, you can alter the contents by pressing 002 if you are in the alter mode (scanning decimal points). To enter the alter-mode, press the ALTER / RST / Ø key.	040 101 002
+ — A short beep; the memory address increments one location.	040 102 XXX
Ø PC 6 OUT AF 2 — A short beep you enter as each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location.	040 103 XXX

Ø SP 1 Ø	A short beep as you enter each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location.	040 104 XXX
Ø DE 4 GO Ø	A short beep as you enter each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location.	040 105 XXX

Continue entering the 3-digit octal numbers in the "Contents" column. The results will be the same as in the previous chart; a short beep as you enter each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location. If you made an error, press the – key and then enter the correct 3-digit octal number.

MEMORY ADDRESS	CONTENTS
040105	006
040106	004
040107	041
040110	170
040111	040
040112	021
040113	013
040114	040
040115	016
040116	011
040117	176
040120	022
040121	043
040122	023
040123	015
040124	302
040125	117
040126	040
040127	016
040130	003
040131	076
040132	377
040133	315
040134	053
040135	000
040136	015
040137	302

MEMORY ADDRESS	CONTENTS
040140	131
040141	040
040142	005
040143	302
040144	112
040145	040
040146	076
040147	062
040150	315
040151	140
040152	002
040153	076
040154	062
040155	315
040156	053
040157	000
040160	076
040161	062
040162	315
040163	140
040164	002
040165	303
040166	105
040167	040
040170	377
040171	262
040172	270

MEMORY ADDRESS	CONTENTS
040173	272
040174	275
040175	377
040176	222
040177	200
040200	377
040201	237
040202	244
040203	377
040204	272
040205	230
040206	377
040207	220
040210	326
040211	302
040212	377
040213	275
040214	272
040215	271
040216	271
040217	373
040220	271
040221	240
040222	377
040223	236
040224	376
040225	362
040226	236
040227	376
040230	362
040231	236
040232	376
040233	362

KEYS PRESSED/RESULT	DISPLAY
ALTER / RST/Ø A short beep; the scanning decimal points will go out.	040 234 XXX
REG ■ A short beep; the left six decimal points light.	X.X.X. X.X.X. XXX
PC 6 OUT A short beep; the decimal points will go out.	XXX XXX Pc
ALTER / RST/Ø A short beep; the left six decimal points will scan from right to left.	XXX XXX Pc
Ø DE 4 GO Ø SP 1 Ø Ø A short beep as you enter each digit and a medium beep as each 3-digit octal number is completed. The decimal points continue to scan.	040 100 Pc
ALTER / RST/Ø A short beep; the scanning decimal points will go out.	040 100 Pc
DE 4 GO A short beep; the "Initial Test Routine" will execute. At the end of the fourth display, the speaker will beep twice and the routine will repeat.	your HB IS up And runn ing ⊢⊣ ⊢⊣ ⊢⊣
Ø MEM # RTM/Ø Press these keys simultaneously. A medium beep and the routine will stop executing.	XXX XXX XXX

MEMORY TEST ROUTINE

The following test routine will exercise a 4K or 8K block of memory originating at 8K (040000). This routine is entered through the front panel and functions as follows: The B register is compared with memory. The HL register pair contains the address being tested. To initialize the routine; 000 is written into all memory locations being tested, the B register is set to 000, and the HL register pair is set to the starting address. The HL register pair is incremented to the ending address and each location is compared with the B register. The B register is then incremented to 001 and each memory location is incremented and compared with the B register. This process continues until the B register is incremented to 377. The process described above then repeats.

If the contents of the memory location that's address is in the HL register pair does not correspond with the value in the B register, the routine will halt, the speaker will "beep", and control will return to the front panel. You can then use the front panel functions to isolate the failure.

NOTE: You cannot test the entire 4K or 8K block of memory with the "Memory Test Routine." Locations 040000 to 040100 are reserved to support the monitor program and the uppermost 80_{10} bytes (057260 to 057377 for 4K and 077260 to 077377 for 8K) are reserved for the stack. Therefore, only locations 040100 to 040160 are occupied by the "Memory Test Routine." Refer to the "Memory Map" under "System Considerations."

The test routine is entered through the front panel. The method of entry and the resulting display will be the same as the "Initial Test Routine." X=random number.

KEYS PRESSED/RESULT	DISPLAY
MEM # RTM/Ø — A short beep; all decimal points light.	XXX XXX XXX
Ø DE 4 GO Ø SP 1 Ø Ø — A short beep as you enter each digit and a medium beep as each 3-digit octal number is completed. When you enter the sixth digit, the decimals points go out.	040 100 XXX
ALTER / RST/Ø — A short beep; the decimal points scan from right to left.	040 100 XXX
Ø DE 4 GO SP 1 — A short beep as you enter each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location.	040 101 XXX

KEYS PRESSED/RESULT	DISPLAY
SP 1 PC 6 OUT 0 — A short beep as you enter each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location.	040 102 XXX
0 DE 4 GO 0 — A short beep as you enter each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location.	040 103 XXX
0 AF 2 SP 1 — A short beep as you enter each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location.	040 104 XXX
AF 2 PC 6 OUT 0 — A short beep as you enter each digit and a medium beep when the 3-digit octal number is completed. The memory address increments one location.	040 105 XXX

Continue entering the 3-digit octal numbers in the "Contents" column. The results will be the same as in the previous chart; a short beep as you enter each digit, a medium beep when each 3-digit octal number is completed. The memory address will increment one location. If you make an error, press the — key and then enter the correct 3-digit octal number.

NOTE: The contents of the next memory location (040 105) depends upon the amount of memory installed on your memory board. If you have a 4K board, use 057. If you have an 8K board, use 077.

MEMORY ADDRESS	CONTENTS
040 105	057(4K) 077(8K)
040 106	066
040 107	000
040 110	315
040 111	147
040 112	040

MEMORY ADDRESS	CONTENTS
040 113	043
040 114	302
040 115	106
040 116	040
040 117	006
040 120	000
040 121	052

MEMORY ADDRESS	CONTENTS
040122	101
040123	040
040124	004
040125	064
040126	176
040127	270
040130	312
040131	135
040132	040
040133	166
040134	000
040135	315
040136	147
040137	040
040140	043
040141	302
040142	125

MEMORY ADDRESS	CONTENTS
040143	040
040144	303
040145	121
040146	040
040147	172
040150	254
040151	300
040152	173
040153	255
040154	311

KEYS PRESSED/RESULT	DISPLAY
A short beep; the scanning decimal points will go out.	040 155 XXX
A short beep; all decimal points light.	040 155 XXX
A short beep as you enter each digit and a medium beep as each 3-digit octal number is completed. When you enter the sixth digit, the decimal points go out.	040 100 041
A short beep; the memory address increments one location.	040 101 160

KEYS PRESSED/RESULT	DISPLAY
Hold down the + key; the memory address will continuously increment. Check the contents to be sure the routine is entered correctly; release the + key at location 040 154.	Continuously increments.
If you find an error, press the ALTER / RST / Ø key and enter the correct 3-digit octal number; press the ALTER / RST / Ø key again to exit ALTER.	XXX XXX Change contents.
A short beep; the left six decimal points light	X.X.X. X.X.X. XXX
A short beep; the decimal points will go out.	XXX XXX Pc
A short beep; the left six decimal points will scan from right to left.	XXX XXX Pc
A short beep as you enter each digit and a medium beep as each 3-digit octal number is completed. The decimal points continue to scan.	040 100 Pc
A short beep; the scanning decimal points will go out.	040 100 Pc
A short beep; the left six decimal points light.	X.X.X. X.X.X. XXX

KEYS PRESSED/RESULT		DISPLAY
BC 3	A short beep; the decimal points will go out.	XXX XXX bC
DE 4 GO	A short beep; the "Memory Test Routine" will execute.	

If the routine executes successfully, the B register (left 3 digits) will increment from 000 to 377. The routine is then complete and may be halted (press the ALTER / RST / Ø and Ø keys simultaneously). When the routine successfully runs from 000 to 377, proceed to the "Operation" section.

If the routine fails to execute, the speaker will sound and the B register (left 3 digits) will display the memory **content** where the test failed. The HL register pair (press REG and then HL 5 IN) will display the memory **address** where the test failed.

Memory failures usually fall into two categories: data and address. A data failure constitutes a particular number or group of numbers from 000 to 377 that cannot be written into or recalled from memory. This type of failure may be due to faulty data buffers, a solder bridge, or defective cells in a memory chip. Since there are eight memory IC's, one for each bit of a byte, it is possible to write a combination of bytes at the address where the test routine failed to determine which, if any, of the memory IC's are at fault. If the memory IC's are interchanged between bits, the difficulty should move with the faulty IC. Be cautious when interchanging memory IC's, since these IC's are MOS devices. The following chart will help you locate each memory IC.

		LOWER 4K	UPPER 4K
LEAST SIGNIFICANT DATA DIGIT	D_0	IC114	IC106
	D_1	IC115	IC107
	D_2	IC116	IC108
	D_3	IC117	IC109
	D_4	IC118	IC110
	D_5	IC119	IC111
MOST SIGNIFICANT DATA DIGIT	D_6	IC120	IC112
	D_7	IC121	IC113

Address faults are the most difficult to isolate. They may be caused by solder bridges between address lines on the circuit board or by a faulty memory IC. When address lines are shorted together (held high or low), the CPU cannot access the memory locations requested. Often, more than one address will access the location. Therefore, recalling how the "Memory Test Routine" functions, you can see that a given memory location will be incremented too often.

If the "Memory Test Routine" fails, try to write the current number in the B register (left 3 digits) into that memory location. If this number can be written into memory, the fault is usually address related.

Although address faults are difficult to locate, a pattern will be evident when you examine all address failures. While displaying the HL register pair, press the

key after you have checked each failure.

The routine will execute to the next failure. The most practical approach for locating an address failure is to substitute memory IC's, one at a time, until you locate the problem.

If the "Memory Test Routine" fails to operate at all, proceed to "Chart 2" in the "Troubleshooting" section.

TROUBLESHOOTING

IMPORTANT NOTICE

All Heath computer hardware and software products were designed to work together as a complete system. Proper operation can be assured only when the computers are used with Heath designed or approved accessories. Heath does not assume the responsibility for improper operation resulting from custom interfacing, custom software, or the use of accessories not approved by Heath Company.

A replacement parts list is provided at the rear of this Manual for your convenience. However, if you attempt to service this unit yourself during the Warranty period, you may void the Warranty.

The troubleshooting information for your Computer is presented in a series of test charts. Read the following paragraphs carefully before you begin troubleshooting.

PRECAUTIONS FOR TROUBLESHOOTING

1. Be cautious when testing transistors and integrated circuits. Although they have almost unlimited life when used properly, they are much more vulnerable to damage from excessive voltage and current than other circuit components.
2. Be careful so you do not short adjacent pins together on the mother board when making voltage measurements. If the probe slips, for example, and shorts pin 1 and pin 2, it may damage one or more components.

3. Do not remove any components or circuit boards while the Computer is turned on.

4. When you make repairs to the Computer, make sure you eliminate the cause as well as the effect of the trouble. If, for example, you find a damaged resistor, be sure you find out what caused the resistor to be damaged. If the cause is not eliminated, the replacement resistor may also become damaged when the Computer is put back into operation.

5. Refer to the "X-Ray Views" and "Schematic Diagram" to locate the various components.

6. The following symbols and procedures are used in the troubleshooting charts:

Follow the "YES" arrow when you obtain the proper result or condition.

Follow the "NO" arrow when you do not obtain the proper result or condition.

This symbol indicates a bus pin connection.

This symbol indicates a wire connection to a circuit board.

- N/O means non-operative.

- Unless called for, pulse width and pulse shape are not measured. Only the excursion between TTL high and TTL low states is important for these tests.

  H=TTL high (+2V minimum)

  L=TTL low (+.8V maximum)

- A logic probe may be used instead of an oscilloscope for all measurements. Where noted, a logic probe is preferred instead of an oscilloscope.

In an extreme case where you are unable to resolve a difficulty, refer to "Service Information" inside the rear cover of this Manual. Your Warranty is located inside the front cover of this Manual.

TROUBLESHOOTING CHARTS

CHART 1

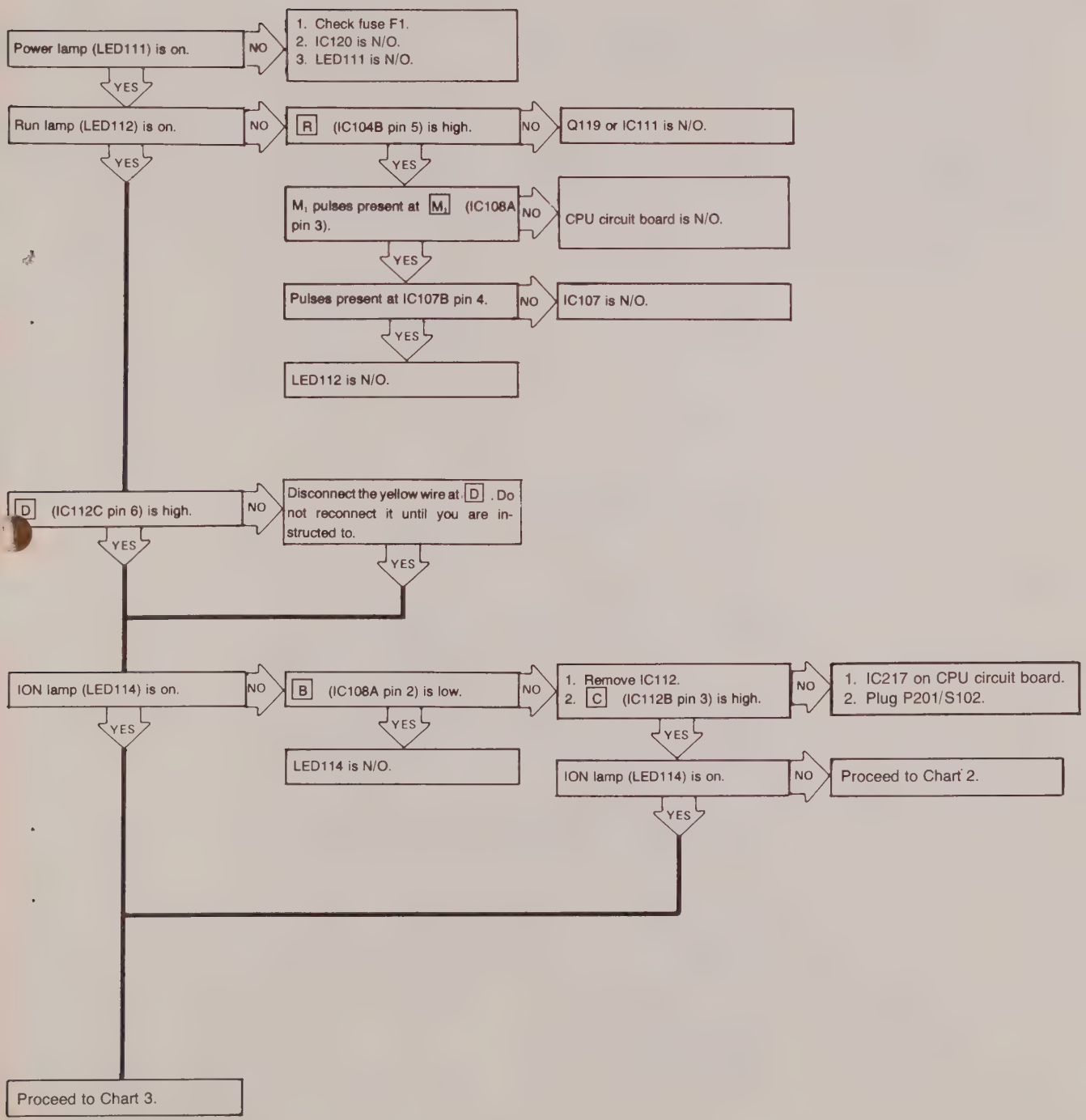


CHART 2

NOTE: Chart 2 pertains only to the Memory Circuit Board.

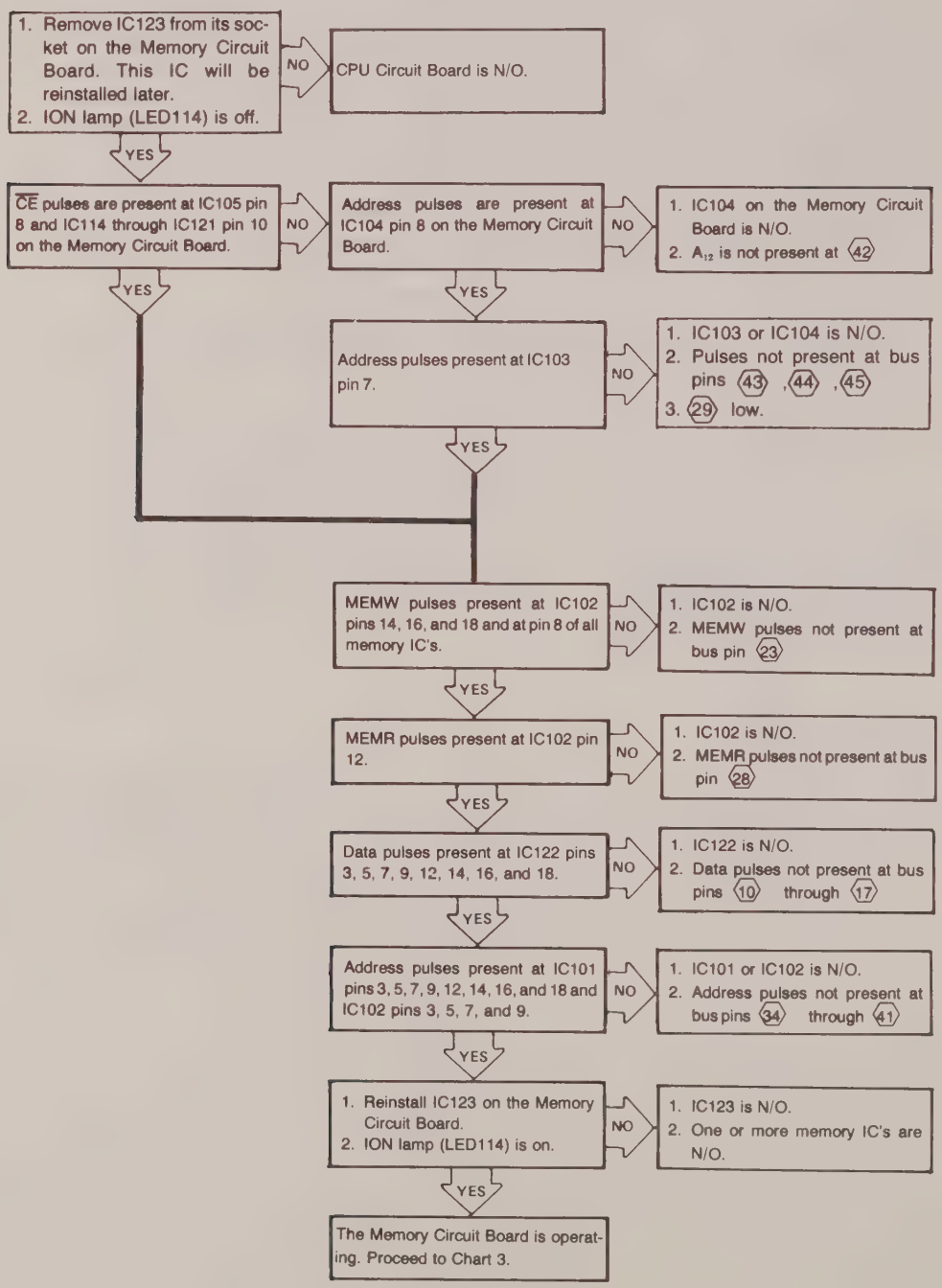

1. Remove IC123 from its socket on the Memory Circuit Board. This IC will be reinstalled later.
2. ION lamp (LED114) is off.
NO
CPU Circuit Board is N/O.
YES
$\overline{CE}$ pulses are present at IC105 pin 8 and IC114 through IC121 pin 10 on the Memory Circuit Board.
NO
Address pulses are present at IC104 pin 8 on the Memory Circuit Board.
NO
1. IC104 on the Memory Circuit Board is N/O.
2. A_{12} is not present at (42)
YES
YES
Address pulses present at IC103 pin 7.
NO
1. IC103 or IC104 is N/O.
2. Pulses not present at bus pins (43), (44), (45)
3. (29) low.
YES
MEMW pulses present at IC102 pins 14, 16, and 18 and at pin 8 of all memory IC's.
NO
1. IC102 is N/O.
2. MEMW pulses not present at bus pin (23)
YES
MEMR pulses present at IC102 pin 12.
NO
1. IC102 is N/O.
2. MEMR pulses not present at bus pin (28)
YES
Data pulses present at IC122 pins 3, 5, 7, 9, 12, 14, 16, and 18.
NO
1. IC122 is N/O.
2. Data pulses not present at bus pins (10) through (17)
YES
Address pulses present at IC101 pins 3, 5, 7, 9, 12, 14, 16, and 18 and IC102 pins 3, 5, 7, and 9.
NO
1. IC101 or IC102 is N/O.
2. Address pulses not present at bus pins (34) through (41)
YES
1. Reinstall IC123 on the Memory Circuit Board.
2. ION lamp (LED114) is on.
NO
1. IC123 is N/O.
2. One or more memory IC's are N/O.
YES
The Memory Circuit Board is operating. Proceed to Chart 3.

CHART 3

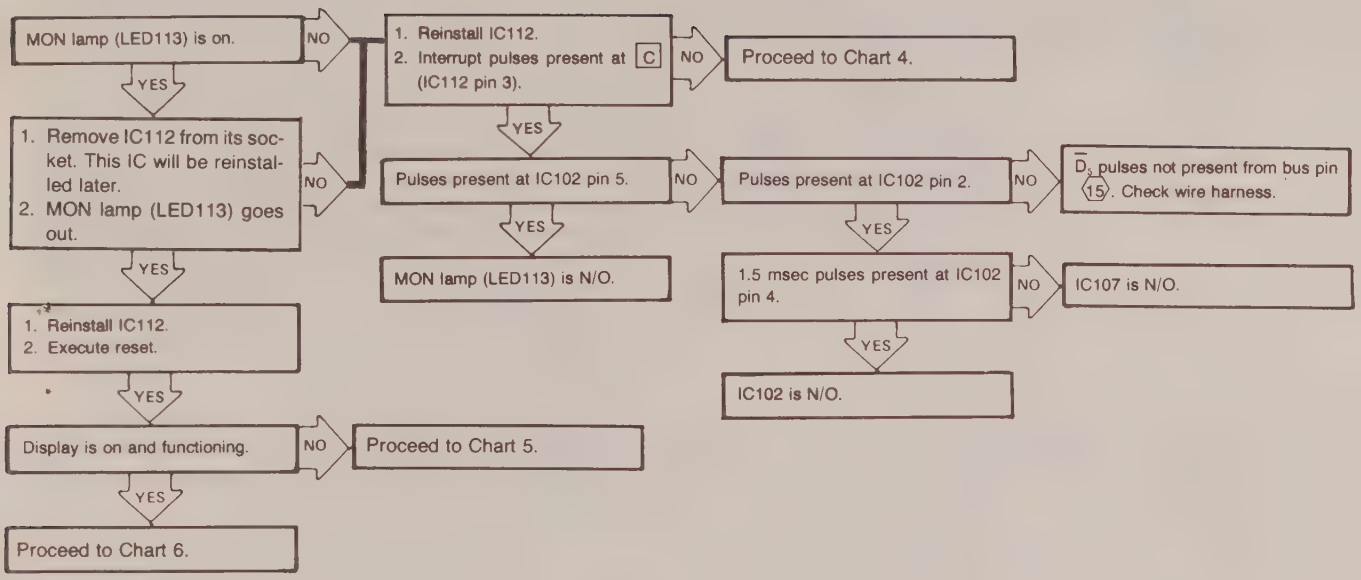

MON lamp (LED113) is on.
NO
YES
1. Remove IC112 from its socket. This IC will be reinstalled later.
2. MON lamp (LED113) goes out.
NO
YES
1. Reinstall IC112.
2. Execute reset.
YES
Display is on and functioning.
NO
Proceed to Chart 5.
YES
Proceed to Chart 6.
1. Reinstall IC112.
2. Interrupt pulses present at C (IC112 pin 3).
NO
Proceed to Chart 4.
YES
Pulses present at IC102 pin 5.
NO
YES
MON lamp (LED113) is N/O.
Pulses present at IC102 pin 2.
NO
$\overline{D}_5$ pulses not present from bus pin 15. Check wire harness.
YES
1.5 msec pulses present at IC102 pin 4.
NO
IC107 is N/O.
YES
IC102 is N/O.

CHART 4

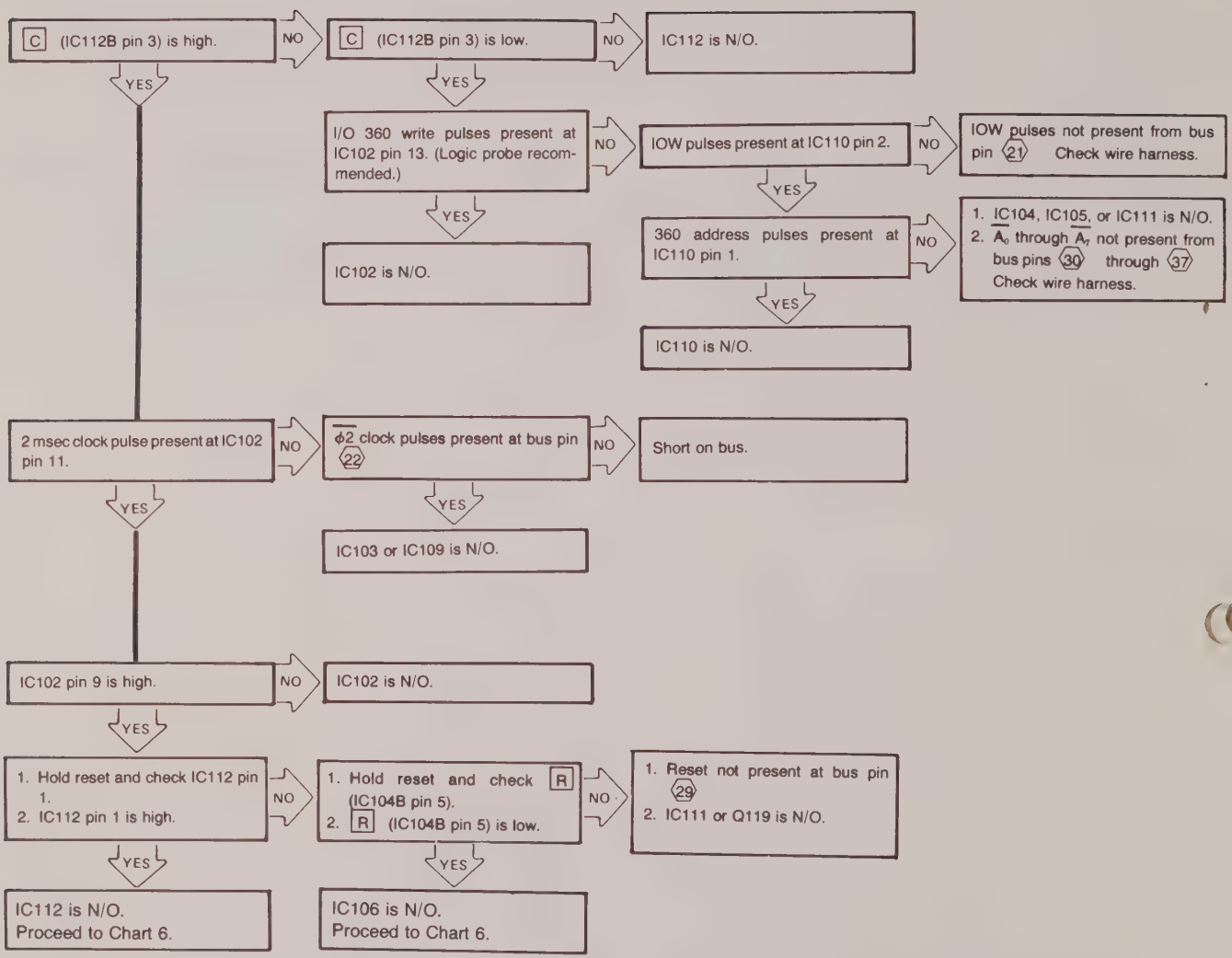

C (IC112B pin 3) is high.
NO
C (IC112B pin 3) is low.
NO
IC112 is N/O.
YES
YES
I/O 360 write pulses present at IC102 pin 13. (Logic probe recommended.)
NO
IOW pulses present at IC110 pin 2.
NO
IOW pulses not present from bus pin 21 Check wire harness.
YES
YES
IC102 is N/O.
360 address pulses present at IC110 pin 1.
NO
1. IC104, IC105, or IC111 is N/O.
2. A0 through A7 not present from bus pins 30 through 37 Check wire harness.
YES
IC110 is N/O.
2 msec clock pulse present at IC102 pin 11.
NO
φ2 clock pulses present at bus pin 22
NO
Short on bus.
YES
YES
IC103 or IC109 is N/O.
IC102 pin 9 is high.
NO
IC102 is N/O.
YES
1. Hold reset and check IC112 pin 1.
2. IC112 pin 1 is high.
NO
1. Hold reset and check R (IC104B pin 5).
2. R (IC104B pin 5) is low.
NO
1. Reset not present at bus pin 29
2. IC111 or Q119 is N/O.
YES
YES
IC112 is N/O.
Proceed to Chart 6.
IC106 is N/O.
Proceed to Chart 6.

CHART 5

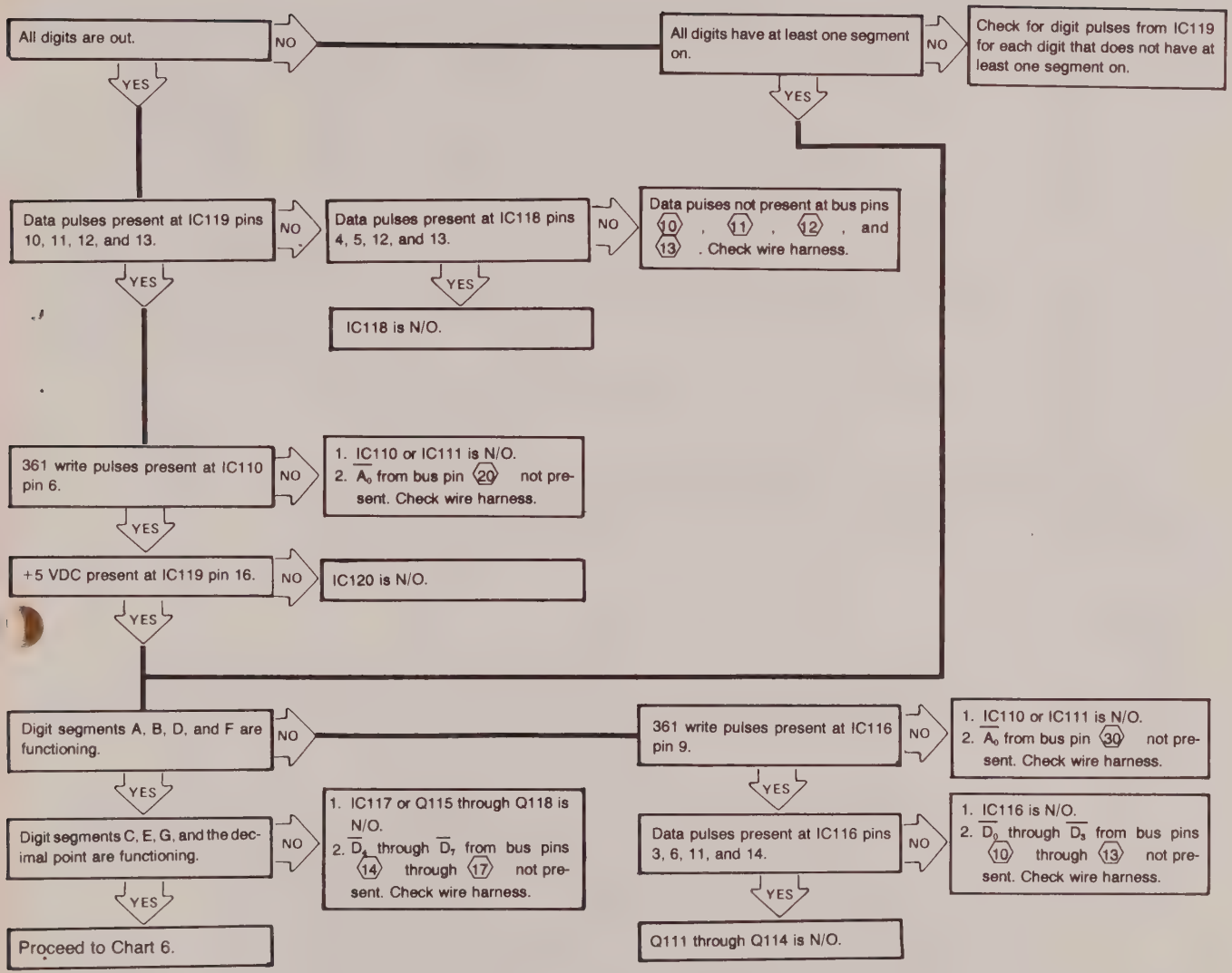
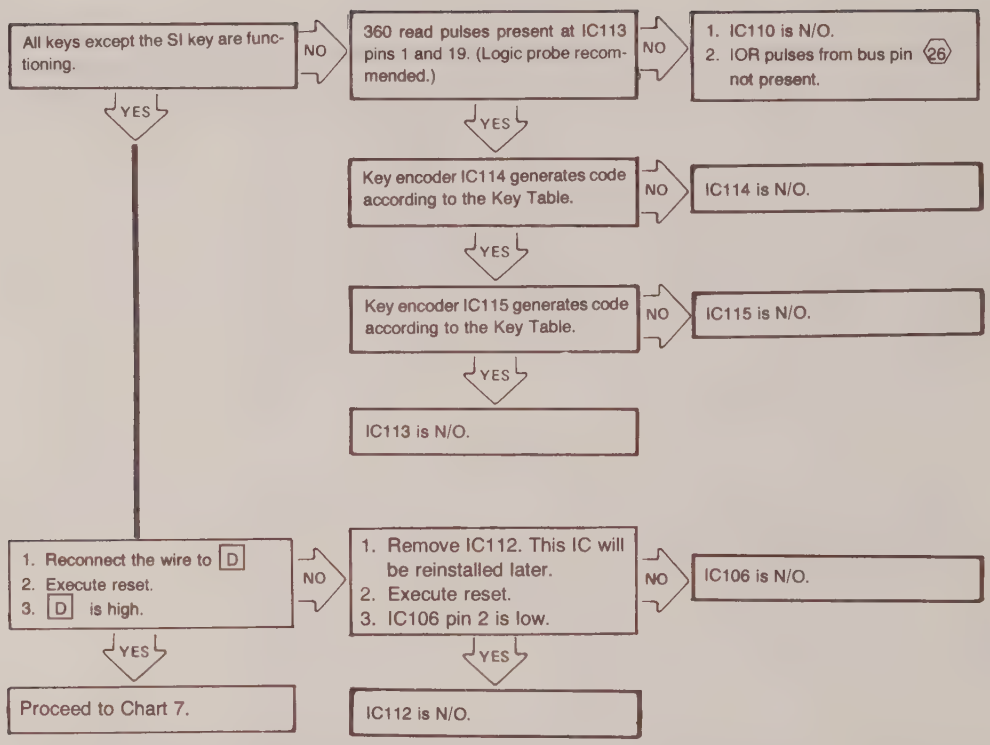

All digits are out.
NO
YES
All digits have at least one segment on.
NO
Check for digit pulses from IC119 for each digit that does not have at least one segment on.
YES
Data pulses present at IC119 pins 10, 11, 12, and 13.
NO
Data pulses present at IC118 pins 4, 5, 12, and 13.
NO
Data pulses not present at bus pins 10, 11, 12, and 13. Check wire harness.
YES
YES
IC118 is N/O.
361 write pulses present at IC110 pin 6.
NO
1. IC110 or IC111 is N/O.
2. $\overline{A_0}$ from bus pin 20 not present. Check wire harness.
YES
+5 VDC present at IC119 pin 16.
NO
IC120 is N/O.
YES
Digit segments A, B, D, and F are functioning.
NO
361 write pulses present at IC116 pin 9.
NO
1. IC110 or IC111 is N/O.
2. $\overline{A_0}$ from bus pin 30 not present. Check wire harness.
YES
YES
Digit segments C, E, G, and the decimal point are functioning.
NO
1. IC117 or Q115 through Q118 is N/O.
2. $\overline{D_4}$ through $\overline{D_7}$ from bus pins 14 through 17 not present. Check wire harness.
Data pulses present at IC116 pins 3, 6, 11, and 14.
NO
1. IC116 is N/O.
2. $\overline{D_0}$ through $\overline{D_3}$ from bus pins 10 through 13 not present. Check wire harness.
YES
YES
Proceed to Chart 6.
Q111 through Q114 is N/O.

CHART 6



The following Key Tables give the output states of IC114 and IC115 when a particular key is pressed. H = TTL high (+2 V minimum) L = TTL low (+.8 V maximum).

KEY TABLES

KEY PRESSED	IC115 PIN			
	6	7	9	14
NONE	H	H	H	H
8	H	H	H	L
9	H	H	L	L
+	H	L	H	L
−	H	L	L	L
*	L	H	H	L
/	L	H	L	L
#	L	L	H	L
■	L	L	L	L

KEY PRESSED	IC114 PIN			
	6	7	9	14
NONE	H	H	H	H
ϕ	H	H	H	L
1	H	H	L	L
2	H	L	H	L
3	H	L	L	L
4	L	H	H	L
5	L	H	L	L
6	L	L	H	L
7	L	L	L	L

CHART 7

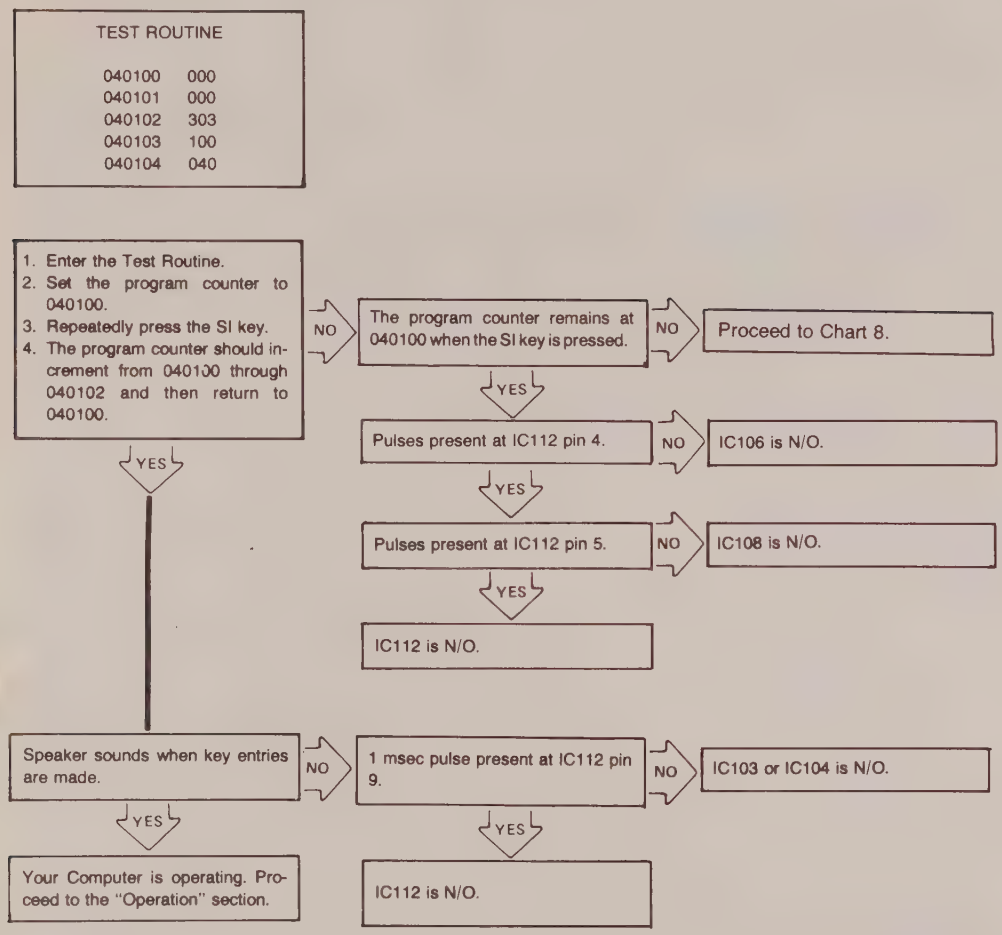
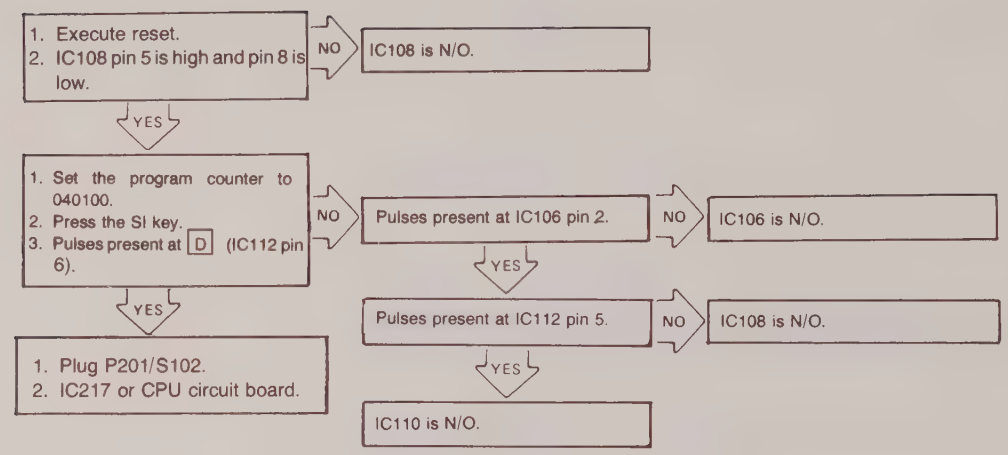

TEST ROUTINE
040100 000
040101 000
040102 303
040103 100
040104 040
1. Enter the Test Routine.
2. Set the program counter to 040100.
3. Repeatedly press the SI key.
4. The program counter should increment from 040100 through 040102 and then return to 040100.
NO
The program counter remains at 040100 when the SI key is pressed.
NO
Proceed to Chart 8.
YES
Pulses present at IC112 pin 4.
NO
IC106 is N/O.
YES
Pulses present at IC112 pin 5.
NO
IC108 is N/O.
YES
IC112 is N/O.
YES
Speaker sounds when key entries are made.
NO
1 msec pulse present at IC112 pin 9.
NO
IC103 or IC104 is N/O.
YES
YES
Your Computer is operating. Proceed to the "Operation" section.
IC112 is N/O.

CHART 8


1. Execute reset.
2. IC108 pin 5 is high and pin 8 is low.
NO
IC108 is N/O.
YES
1. Set the program counter to 040100.
2. Press the SI key.
3. Pulses present at D (IC112 pin 6).
NO
Pulses present at IC106 pin 2.
NO
IC106 is N/O.
YES
YES
Pulses present at IC112 pin 5.
NO
IC108 is N/O.
1. Plug P201/S102.
2. IC217 or CPU circuit board.
YES
IC110 is N/O.

OPERATION

This section of the Manual explains features and basic functions of your H8 Digital Computer. An in-depth discussion on exactly how to perform the various front panel functions in writing a program is contained in the H8 Software Reference Manual.

INTRODUCTION

Refer to Pictorials 6-3 (Illustration Booklet, Page 3) and 6-4 (Page 29) while you read the following introduction.

The Computer front panel contains four status lamps, nine LED displays, and a 16-key keyboard.

Each of the four status lamps, when lit, indicates the status of the Computer, as follows:

ION — Indicates the CPU is accepting interrupts.

MON — Indicates you have control from the front panel.

RUN — Indicates the CPU is in a run condition.

PWR — Indicates that +5 volts is present at the front panel.

The six left LED's display the octal ADDRESS. The three left digits display the high-order address, while the three center digits display the low-order address. The three right digits function as the DATA/REGISTER displays. In the Memory Mode, the three right digits display the data contained at the displayed address. In the Register Mode, the three right digits alphanumerically display the register you are addressing and the six left digits display the register contents.

The keyboard will be described under "Use of the Front Panel."

An important internal feature of the Computer is an audio alert circuit to signal you when you have made an error or successfully entered data. This audio alert circuit has three basic functions:

1. A long beep to verify a Master Reset or an illegal command.

2. A medium beep to verify successful address or data byte completion.

3. A short beep to verify key stroke.

This feature is very useful when you are entering a long program through the front panel. By listening to the feedback signal, you can determine if the Computer has accepted your data.

Another important feature is the Computer's ability to determine the amount of memory installed upon initial power-up. It will do this without destroying the contents of the memory. A detailed discussion of this feature is contained in the Software Reference Manual.

Because the front panel requires a small amount of support RAM, you must be careful when entering data in the first 100 octal locations, starting at the octal address 040000. You will find these locations useful when using the front panel as an I/O device. A discussion on exactly how to access the front panel is contained in the Software Reference Manual.

MODES OF OPERATION

The front panel has the following modes of operation:

Register Mode — Allows you to address any of the 8080's internal registers and read or alter the data in them.

Memory Mode — Allows you to address any location in memory and read or alter the data in that memory location.

Cancel — Allows you to return to the Command Mode.

Load — Allows you to take the data from auxiliary storage and place it into memory. This feature includes a CRC (Cyclic Redundancy Check) check of the data being loaded to assure the detection of bad loads.

Dump — Allows you to take data from memory and place it into auxiliary storage. The data being dumped is dumped with CRC for future loading.

Increment and Decrement — Allows you to increment or decrement one memory address or one register pair. This feature is very useful when stepping through a program.

Go — Allows you to execute a program from the memory location being pointed to by the program counter.

Single Instruction — Allows you to execute the single instruction being pointed to by the program counter.

Reset — Allows you to execute a hardware reset of the CPU registers and serial or parallel interfaces. This reset is the same reset that occurs at power-up. So you do not accidentally execute a reset, you are required to press both the ϕ key and the RST/ϕ key. You should avoid using this reset because you will destroy any information in the CPU registers and serial or parallel ports. If reset is executed, you will have to reinitiate your I/O ports.

Return to Monitor — Allows you to stop the program being executed and return control to the front panel monitor. This does not destroy information in the CPU registers or the serial or parallel ports. You will find this feature useful in returning from a program which may be in a loop. To execute a return to monitor, press both the ϕ key and the RTM/ϕ key.

Input — Allows you to input data from an I/O port and display it on the front panel.

Output — Allows you to output data from the front panel to an I/O port.

Auto-Repeat — Allows you to hold down any key on the front panel, thereby repeating the corresponding operation or entry automatically at a 2.5 Hz rate. This allows you to enter or execute programs using fewer key strokes. It is especially useful when stepping through memory or executing single instructions.

Input and output commands are very useful in setting up and initializing an I/O port, such as UART's, that require a setup word.

DECIMAL POINT OPERATION

Lit decimal points on the LED displays verify that the Computer is ready for data input and that the keys perform their numeric or register function. If the left six decimal points are lit, information about a register will be accepted. If all of the decimal points are lit, information for memory reference will be accepted.

When the decimal points are lit constantly, you can enter a memory location. When the decimal points are scanning from right to left, any key entries will alter that location.

Four decimal point displays can be obtained. These displays and their functions are:

1. All decimal points lit constantly — ready to select a memory address.

2. Left six decimal points lit constantly — ready to select a register.

3. All decimal points scanning — ready to alter memory contents.

4. Left six decimal points scanning — ready to alter register contents.

SPLIT OCTAL DISPLAY

The 7-segment LED displays on the front panel of your H8 display the address and data information in octal format. This is accomplished by grouping three binary bits together and displaying the equivalent octal value. An example of this octal format is shown below.

BINARY NUMBER ll lll lll

OCTAL NUMBER

Notice that the left digit represents only two bits of the binary number, whereas the two right digits represent three bits each of the binary number. Therefore, the largest octal number the left digit can display is 3, while the two right digits can each display 7. Thus, for an 8-bit binary word, the maximum octal value is 377.

16-bit words are displayed in split octal format. In this type of format, 16-bit words are treated as two 8-bit words. Therefore, for a 16-bit binary word, the maximum octal value is 377 377. Notice that maximum value for each 3-digit octal number is 377.

USE OF THE FRONT PANEL

REGISTER MODE

To enter the Register Mode:

1. Press the REG (register) key. The decimal points in the six left digits will light.

2. Press the desired register pair key. The selected register pair will be displayed in the two right digits. The contents of the register pair will be displayed in the six left digits, three digits per register, displayed octally.

To alter the contents of the register pair:

1. Press the ALTER key. The six left decimals will scan from right to left, verifying that you are in the alter Mode.

2. Enter the six octal digits of the new data. The data will be entered from right to left.

   - Be sure you enter leading zeros so that you always enter six digits.

   - You will hear a short beep as each digit is entered and a medium beep when each 3-digit octal number is completed.

   - If you do not want to change the contents of the first register of the pair, enter the same number displayed in that register.

   - If only the contents of the first register of the pair is to be changed, press the CANCEL key followed by the ALTER key at the completion of the first three digits. The scanning decimals will go out, verifying that you are out of the Alter Mode.

MEMORY MODE

To enter the Memory Mode:

1. Press the MEM (memory) key. All of the decimals will be displayed. The address will be displayed in the six left digits and the data of that address will be displayed in the three right digits.

To alter the address:

1. Enter the six octal digits on the high-and-low order address byte you want.

   - Enter the 3-digit octal number for the high-order address byte first.

   - Enter the 3-digit octal number for the low-order address byte second.

   - Be sure you enter leading zeros so that you always enter six digits.

   - You will hear a short beep as each digit is entered and a medium beep when each 3-digit octal number is completed.

   - Upon entering the sixth digit, the decimals will go out and the monitor program will return to the memory display mode.

   - The left three digits display the high-order address byte and the center three digits display the low-order address byte. The right three digits display the data at the selected memory address.

HIGH ORDER ADDRESS LOCATION | LOW ORDER ADDRESS LOCATION | DATA AT LOCATION 040 100

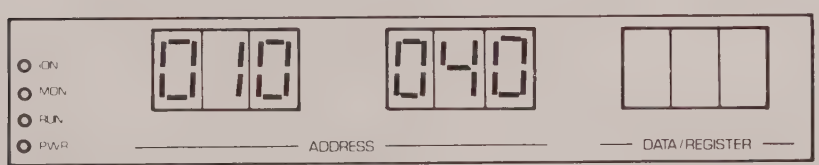
FRONT PANEL MEMORY DISPLAY

HIGH ORDER CONTENTS | LOW ORDER CONTENTS | REGISTER IDENTIFICATION

FRONT PANEL REGISTER DISPLAY (PROGRAM COUNTER SELECTED)

CONTENTS OF THE A REGISTER | CONTENTS OF THE F REGISTER | REGISTER IDENTIFICATION

FRONT PANEL REGISTER DISPLAY (AF REGISTER PAIR SELECTED)



DATA | PORT NUMBER

I/O PORT DISPLAY

PICTORIAL 6-4

To alter the data at the displayed address:

1. Press the ALTER key. The decimals will scan from right to left, verifying that you are in the Alter Mode.

2. Enter the desired octal number on the keyboard.

   - The new data will be entered from right to left.

   - Upon entering the third digit, the address will automatically be incremented one memory address.

3. Press the ALTER key. The scanning decimals will go out, verifying that you are out of the Alter Mode.

To check the new data entered:

1. Press the − (minus) key. The address will decrement one memory address.

2. Press the + (plus) key. The address will increment one memory address.

   - The address will continue to increment or decrement as long as you press the + (plus) or − (minus) key.

CANCEL

If you enter a mode or digit by mistake:

1. Press the CANCEL key.

   - You will hear a long beep when you press the CANCEL key, verifying that the remainder of the operation has been cancelled.

2. Start the process over again from the beginning, including re-entering the desired mode.

LOADING AND DUMPING DATA

To load or dump data, you must have a serial or parallel card along with a load/dump device assigned to I/O port address 370-371. Since these addresses are fixed in ROM, the load/dump ports cannot be reassigned.

Dumping (Storing) Data

To dump data from memory:

1. Enter the low order byte of the starting address of the data to be dumped at memory location 040000.

2. Enter the high order byte of the starting address at memory location 040001.

3. Enter the entry point address in the PC register.

   - The entry point address is where you want the program to begin execution when you load the data back into memory in the future.

4. Return to the Memory Mode and enter the ending address of the data to be dumped.

   - The ADDRESS displays will read the ending address.

   - Disregard the DATA displays at this time.

5. Press the DUMP key. The dump device will start.

   - When using the cassette interface, there will be a delay in the dump process to allow the tape machine to come up to speed.

   - The ADDRESS displays will show the starting address of the data being dumped and will increment to the ending address.

   - The DATA displays will flash with the data being dumped. You will hear a long beep at the end of the dump, verifying completion of this operation.

   - The ADDRESS displays will stop at the ending address of the stored data.

Loading Data

To load data:

1. Ready the loading device.
2. Press the LOAD key. The load device will start.
   - There will be a delay in the data being loaded because of the leader on the tape and the sync characters being read.
   - When the leader is complete, the ADDRESS displays will show the starting address and increment to the ending address.
   - The DATA displays will flash with the data being loaded. The incoming data is checked by CRC.
   - At the end of a successful load, you will hear a long beep to verify that the load has been successful.
   - If the load is unsuccessful, you will hear a pulsing beep to signify a bad load. Therefore, you must repeat the load procedure.

EXECUTING A SAVED PROGRAM

To execute a program that has been dumped with the program counter (entry point) saved, and then reloaded:

1. Press the GO key.

   If you did not save your program starting address, you must enter the starting address in the program counter before pressing the GO key.

INPUTTING AND OUTPUTTING

To set up and initialize the I/O ports, refer to your programming card or specific manuals for information to be output.

Inputting From a Port

To input from a port:

1. Press the MEM (memory) key.
2. Enter three zeros followed by the 3-digit address of the port you want to input from.
   - The ADDRESS displays will show 000AAA (AAA is the port address).
3. Press the IN key.
   - The three left digits will display the data at the port.
   - Disregard the DATA/REGISTER displays.
4. To continue inputting, press the IN key.
   - You do not have to re-enter 000 AAA.

Outputting to a Port

To output to a port:

1. Press the MEM (memory) key.
2. Enter the three digit data byte you want to output, followed by the 3-digit port address you want to output to.
   - The ADDRESS displays will show DDDAAA (DDD is the data byte and AAA is the port address).
3. Press the OUT key.
   - The data will be output to the desired port address.
   - Disregard the DATA/REGISTER displays.
4. To continue outputting, press the OUT key.
   - You do not have to re-enter DDD AAA.

Once a port is initialized, you may input data from that port and then output the same data to the same port to check for proper port operation. This process is known as an echo check.

THEORY OF OPERATION

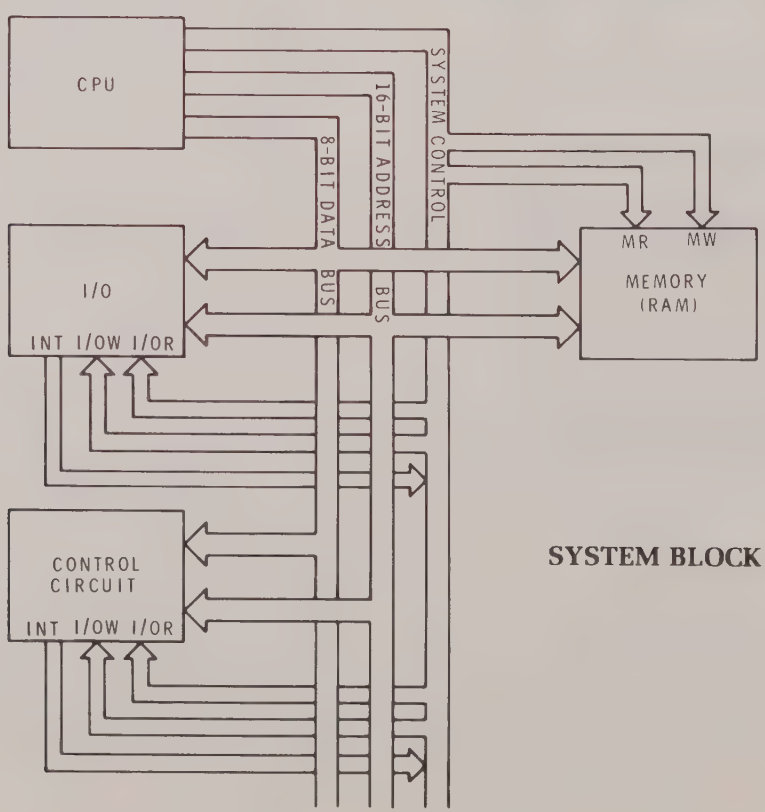
SYSTEM DESCRIPTION

Refer to the System Block Diagram while you read this System Description.

The basic H8 Digital Computer consists of a central processing unit (CPU), an input/output (I/O) device, random access memory (RAM), and a 50-pin bus. You can expand the H8 by adding additional I/O devices for communications with peripherals such as video terminals, hard-copy machines, and tape readers.

The CPU contains an on-board, read-only memory (ROM) programmed to communicate with the front panel and additional I/O devices. This eliminates the need to hand load a boot program upon initial power-up of the H8.

The 50-pin bus contains sixteen address lines, eight bi-directional data lines, five of the CPU's available interrupt lines, and the system control lines. The four basic control signals are the I/O read, I/O write, memory read, and memory write. A read signal indicates information going from a device to the CPU and a write signal indicates information going from the CPU to a device.



SYSTEM BLOCK DIAGRAM

CPU CIRCUIT

Refer to the CPU Block Diagram (Illustration Booklet, Page 4) and the Schematic Diagram (fold-in) while you read the Theory of Operation.

This Theory of Operation will describe the general operation of the CPU circuit. For a detailed description of the 8080A microprocessor, refer to "The 8080 Central Processor Unit" section of the "Theory of Operation."

The heart of the CPU circuit is the 8080A microprocessor (IC213). Because the microprocessor is a dynamic device, it must have an external clock. Clock generator IC212 generates the 2-phase, nonoverlapping, 2.048 MHz clock signal.

PD_0 through PD_7, the 8-bit bi-directional data bus, contains both incoming and outgoing data, along with the 8080A status word. The status word appears on the data bus at the end of T_1 of all machine cycles. A negative going signal on the status strobe (STSTB) line from the clock generator (IC212) signals the system controller (IC214) that the data on the data bus at this time is the status word. The status word is then removed from the data bus and decoded into the following control signals: memory read (MEMR), memory write (MEMW), I/O read (IOR), I/O write (IOW), and interrupt enable (INTE). These control signals are then buffered and inverted by the read/write buffer (IC208A) and presented on the data bus for system control.

The 8080A microprocessor has a 16-line address bus, A_0 through A_{15}, supplying 65,536 possible memory locations. These address signals are buffered and inverted by address buffers IC205 and IC206 to supply ample drive to the bus. Address lines A_{10} through A_{15} are decoded by address decoder IC207. If all the inputs of IC207 are high, indicating an address below 1024, and a MEMR pulse is present at pin 11 of IC207, the ROM (IC204) will be enabled and the Data In buffer (IC211) will be disabled. The data on the data bus of the system controller (IC214) at this time is then determined by the specific address in ROM being selected by address lines A_0 through A_{10}. This address decoding process is the same process that occurs on each memory board when you read from RAM into the microprocessor. When the address bus is indicating any address above 1025, the ROM is disabled and its output is in its high impedance state to prevent interference on the data bus.

MEMORY WRITE CYCLE

When the microprocessor executes an instruction to store an 8-bit word in memory, the following sequence occurs:

1. The system controller (IC214) examines the status word and determines that a memory write cycle is occurring.

2. The memory write control line (pin 26) of the system controller (IC214) goes low and all other control lines remain high.

3. The memory write signal is then inverted and buffered by the read/write buffer (IC208A) before going onto the bus to enable memory.

4. The lack of a memory read pulse at pin 1 of IC215 disables the Data In buffer (IC211) and enables the Data Out buffer (IC210).

5. The high-order address lines (A_{13} through A_{15}) are decoded, selecting the specific memory board addressed.

6. The low-order address lines (A_0 through A_{12}) are decoded, thus selecting the specific word location on the selected memory board.

7. Once the memory board and location are selected, the MEMW (memory write) pulse allows the information on the data bus to be written into that memory location.

MEMORY READ CYCLE

When the microprocessor executes an instruction to retrieve an 8-bit word from memory, the following sequence occurs:

1. The system controller (IC214) examines the status word and determines that a memory read cycle is occurring.

2. The memory read control line (pin 24) of the system controller (IC214) goes low and all other control lines remain high.

3. The memory read signal is then inverted and buffered by the read/write buffer (IC208A) before going onto the bus to enable memory.

4. A memory read pulse at pin 1 of IC215 enables the Data In buffer (IC211) and disables the Data Out buffer (IC210).

5. The high-order address lines (A_{13} through A_{15}) are decoded, selecting the specific memory board addressed.

6. The low-order address lines (A_0 through A_{12}) are decoded, selecting the specific word location on the selected memory board.

7. Once the memory board and location are selected, the MEMR (memory read) pulse allows the information on the data bus to be read by the microprocessor.

I/O CYCLE

The input/output (I/O) cycle enables the microprocessor to receive data or output data. I/O write and I/O read cycles are very similar to memory write and memory read cycles. When the I/O cycle status word is decoded by the system controller, the I/O control lines are used to enable the input or output device. Only the low-order address lines (A_0 through A_7) are decoded for I/O instructions, giving you 256 decimal locations for input or output ports. NOTE: The top 30 I/O locations are reserved for system software. You should avoid writing programs that address these locations.

INTERRUPTING THE PROCESSOR

While the CPU is in the process of executing a program, it may be called upon to service an input or output device before any information is lost. To do this, the microprocessor must be interrupted, allowing it to store the results of any computations and its place in the current program. Interrupts are executed by taking any of the seven inputs of the interrupt decoder (IC217) to logic 0. When pin 14 of the interrupt decoder goes low, it drives the output (pin 5) of IC209D high. This signals the microprocessor that an interrupt has occurred. The microprocessor finishes its current instruction and sends a status word to the system controller (IC214). The system controller decodes the status word as an interrupt and enables the interrupt buffer (IC218). At this time, an 8-bit status word is presented on the data bus and is decoded by the microprocessor (IC213).

Only bits 3, 4, and 5 of the 8-bit word are variable, depending upon which of the seven interrupt lines were pulled low. The variable bits indicate which of the seven memory locations (10, 20, 30, 40, 50, 60, or 70) the microprocessor should address for instructions on servicing that interrupt. Bits 0, 1, 2, 6, and 7 of the 8-bit word are preset and indicate a restart instruction.

Interrupt 0 is used as a general reset interrupt. Upon initial power-up, the CPU will receive an interrupt 0, causing it to execute a general power-up procedure. Interrupt 10 is used to service the control circuit board and interrupt 20 is used to implement the single instruction mode. When an interrupt is enabled, the outputs of the interrupt decoder (IC217) are set according to the chart on Page 35.

Several features have been incorporated in the CPU for future expansion and flexibility. These features will be explained in the following paragraphs.

HOLD ACKNOWLEDGMENT

To provide for direct memory access (DMA), the CPU circuitry must be completely disconnected from the system bus. This is done by using the hold feature of the 8080A microprocessor.

INTERRUPT DECODER IC217

INTERRUPT	A_2 (pin 6)	A_1 (pin 7)	A_0 (pin 9)	G_S (pin 14)
0 (pin 10)	1	1	1	L (low)
10 (pin 11)	1	1	0	L
20 (pin 12)	1	0	1	L
30 (pin 13)	1	0	0	L
40 (pin 1)	0	1	1	L
50 (pin 2)	0	1	0	L
60 (pin 3)	0	0	1	L
70 (pin 4)	0	0	0	L
no interrupts	1	1	1	H (high)

When the CPU receives a hold request (IC213 pin 13), the hold acknowledge line (HLDA, pin 21) goes high after a brief delay. The HLDA is used to disable address, data, and control buffers so an external device can communicate with memory. IC216B is used as a delay latch to allow sufficient time to complete the current cycle before disabling the data buffers.

The HLDA circuitry works as follows:

1. The HLDA line (IC213 pin 21) at logic 0 prevents the delay latch (IC216B) from being clocked by the $\phi 2$ input (pin 11).

2. When the HLDA line (IC213 pin 21) goes high, the delay latch (IC216B) is clocked at the end of the next $\phi 2$ cycle, setting the Q output (pin 9) high and the $\overline{Q}$ output (pin 8) low.

3. A high at the Q output (pin 9) of the delay latch (IC216B) will disable the address buffers (IC205 and IC206) and the read/write buffer (IC208A).

4. A low at the $\overline{Q}$ output (pin 8) of the delay latch (IC216B) will disable the Data Out buffer (IC210).

IC209C is used to drive the bus with the HLDA signal. The polarity of the HLDA signal is determined by jumpers J1, J2, and J3.

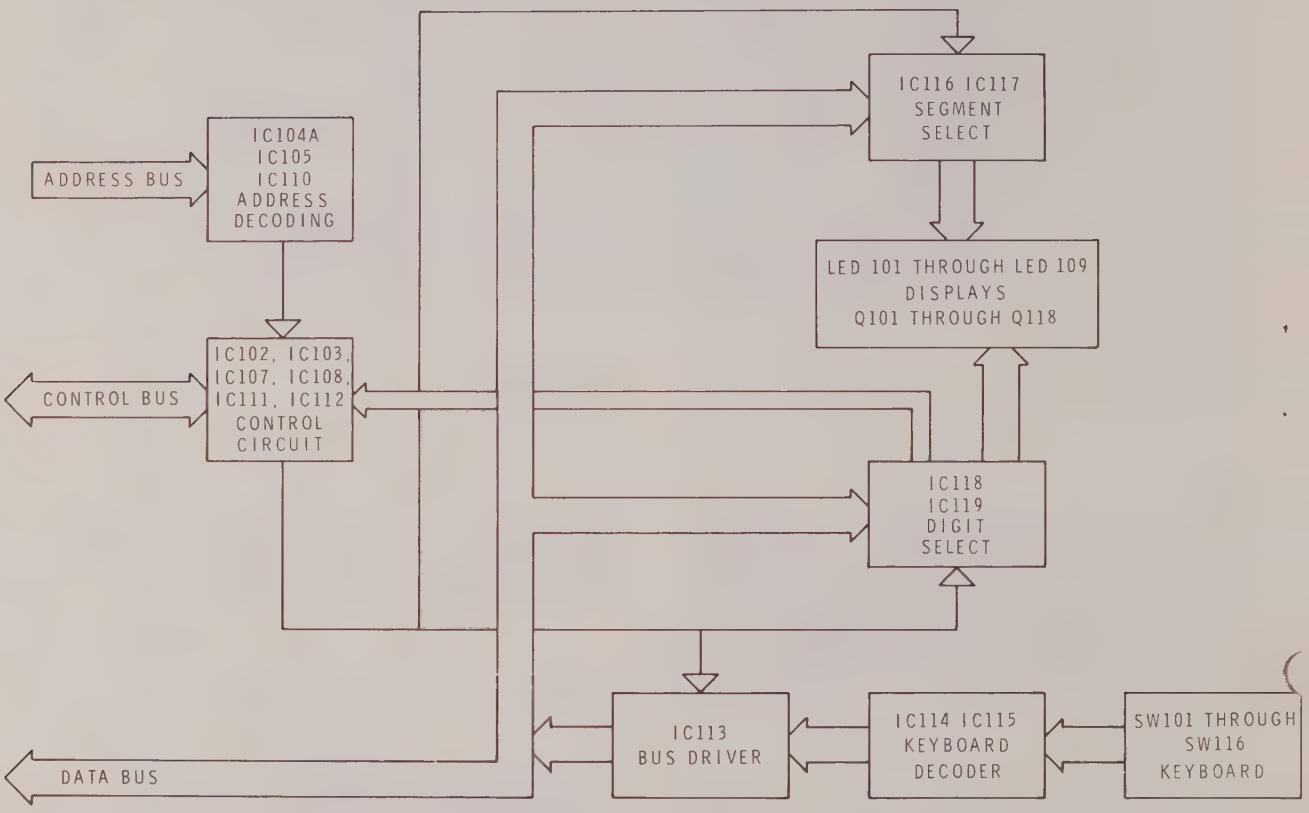
M_1 CYCLE

The M_1 cycle, generated on the CPU circuit board, is the first machine cycle of an instruction.

The microprocessor (IC213) generates a sync pulse at pin 19 during T_1 and T_2 of each machine cycle. This sync pulse is coupled to pin 5 of the clock generator (IC212) where it is gated with the $\phi 1$ signal to produce the $\overline{\text{STSTB}}$ pulse (pin 7). The $\overline{\text{STSTB}}$ (status strobe) pulse indicates to the system controller the presence of a status word on the data bus. Because the M_1 signal at PD5 of the data bus is so narrow (approximately 50 nsec), it is stretched by combining it with the $\overline{\text{STSTB}}$ pulse (pin 3) and the sync pulse (pin 1) at IC216A.

When the D input (pin 2) of IC216A goes high, along with the clear input (pin 1) indicating an M_1 cycle, the leading edge of the next $\overline{\text{STSTB}}$ pulse (pin 3) will clock IC216A. The $\overline{Q}$ output (pin 6) will then go to logic 0. When the sync pulse (pin 1) returns to logic 0, IC216A will be cleared and the $\overline{Q}$ output (pin 6) will return to logic 1, indicating the end of the M_1 cycle. The M_1 signal is inverted and buffered by IC208B and coupled to the control circuit board.

FRONT PANEL (CONTROL CIRCUIT)



CONTROL BLOCK DIAGRAM

Refer to the Control Circuit Block Diagram and the Schematic Diagram while you read this description.

The front panel is an I/O device assigned to address 360 and 361. Information is output to the 7-segment displays and input from the keyboard under control of the ROM (on the CPU circuit board). The ROM contains the appropriate software to service the front panel.

Because the displays are multiplexed, they must be continuously refreshed under CPU control. The front panel generates a level 10 interrupt when it is ready to be refreshed. This interrupt is generated on the front panel by dividing the $\phi 2$ clock. IC103 divides the 2 MHz clock by 4096, producing a pulse every 2 msec.

This pulse is coupled through an inverter (IC109F) to the clock input (pin 11) of IC102B. IC102B holds the interrupt until the CPU has responded to this interrupt. Upon receiving the clock pulse at IC102B pin 11, the $\overline{Q}$ output (pin 9) will go high. The $\overline{Q}$ output of IC102B is gated together with the $\overline{Q}$ output (pin 6) of IC106 at IC112B. At this time, the output (pin 3) of IC112B goes low, interrupting the microprocessor. Once the microprocessor has received the interrupt signal, it will generate an address on the bus according to the following table.

FRONT PANEL ADDRESS 360 AND 361

	$\overline{A}_7$	$\overline{A}_6$	$\overline{A}_5$	$\overline{A}_4$	$\overline{A}_3$	$\overline{A}_2$	$\overline{A}_1$	$\overline{A}_0$
360	0	0	0	0	1	1	1	1
361	0	0	0	0	0	1	1	0

These address signals are decoded by IC104A, IC105, and IC111A and B. They are then gated together with the I/O read and I/O write signals at IC110.

The first address signal generated by the service routine (contained in ROM) is a 360 write signal. A 360 write signal causes pin 3 of IC110B to go low, clocking IC106 and IC118 and latching the high-and low-order data bits from the data bus. Once data is latched in IC118, it is decoded by the digit select decoder (IC119). IC119 selects one of nine LED's to be turned on.

The 360_8 write signal also clears IC102B, removing the interrupt signal from the CPU, and clocks IC102. When IC102A is clocked, it latches the D_5 bit of the data bus, causing the Q output (pin 5) of IC102A to go low, turning on the MON lamp (LED113). The MON lamp indicates that the front panel is being serviced by the CPU.

The second address signal generated is the 361_8 signal at the output (pin 6) of IC110A. When this output goes low, the segment select decoders (IC116 and IC117) will latch the data from the eight bits of the data bus. This data is then driven directly to the seven segments and the decimal point of the selected LED. The segment select decoders determine the value to be displayed on the LED's.

To insure that only the proper segments are turned on in any selected display, the segment select decoders (IC116 and IC117) must be cleared prior to selecting any digit. The 2 msec clock, used to generate the interrupt, is coupled to pin 10 of IC107. IC107 is a monostable with a time constant of approximately 1.5 msec. Each time a new digit is selected, IC107 will be triggered, enabling the segment select decoders (IC116 and IC117). IC107 allows the data to remain in the decoders approximately 1.5 msec before resetting them. This is just prior to a new interrupt being generated, so that as new digits are selected, the segment select decoders will have been cleared by IC107. As IC107 times out and pin 5 returns to logic 0, the Q output (pin 5) of IC102 will go high, turning off the MON lamp. Each time this interrupt is generated, the CPU checks to determine how many of the nine digits have been serviced and stores this information in a scratch pad location for ROM. The ROM program will continue to update all nine displays, one display every 2 msec.

Upon servicing the ninth digit, the ROM program will generate an address decoded as a 360_8 read signal, causing pin 11 of IC110D to go low, thus turning on the data buffer (IC113). At this time, the data on the data bus will be a function of the key pressed. IC114 and IC115 decode the keys according to the following tables.

IC114

KEY	$\overline{A}_3$	$\overline{A}_2$	$\overline{A}_1$	$\overline{A}_0$
ϕ	H	H	H	L
1	H	H	L	L
2	H	L	H	L
3	H	L	L	L
4	L	H	H	L
5	L	H	L	L
6	L	L	H	L
7	L	L	L	L
No key	H	H	H	H

IC115

KEY	$\overline{A}_3$	$\overline{A}_2$	$\overline{A}_1$	$\overline{A}_0$
8	H	H	H	L
9	H	H	L	L
+	H	L	H	L
–	H	L	L	L
*	L	H	H	L
/	L	H	L	L
#	L	L	H	L
■	L	L	L	L
No key	H	H	H	H

The CPU spends approximately 200 μsec updating the front panel, or approximately 10% of its total process time.

MISCELLANEOUS CONTROLS

The CPU circuit generates an M_1 pulse at the beginning of each machine cycle. This M_1 signal is used to trigger a pulse-stretching monostable (IC107) and turn on the RUN lamp. The RUN lamp will remain on as long as the M_1 cycle is generated (every 4.7 msec). An M_1 pulse will not be generated if the CPU is in a hold or wait cycle.

The interrupt enable line from the CPU is used to turn on the ION lamp. This interrupt signal is also used to enable the single instruction divide-by-four counters (IC108A and B).

When the interrupt enable line is low, IC108A and B will count two M_1 cycles before the $\overline{Q}$ output (pin 11) goes low. If, during the update process, D_5 set the latch (IC106), the Q output (pin 2) will be high, enabling IC112C. When the $\overline{Q}$ output (pin 11) of IC108 goes low, pin 6 of IC112C will go low. Pin 6 of IC112 going low is decoded as a level 20 interrupt. Execution of additional instructions is halted unless the single instruction key is pressed. The CPU will not execute additional instructions, but will continue to update the front panel and strobe the keys for additional instructions.

AUDIO ALERT

The audio alert signal is produced by dividing the $\phi 2$ clock signal by 2048, producing approximately a 1 kHz signal. This signal is gated together with the Q output (pin 10) of the data latch (IC106). When the Q output is high, the audio signal is coupled to the speaker. When the Q output is low, IC106 is disabled and the speaker is turned off. Data bit D_7 controls the speaker and is latched by the front panel service routine when selecting a display digit.

RETURN TO MONITOR

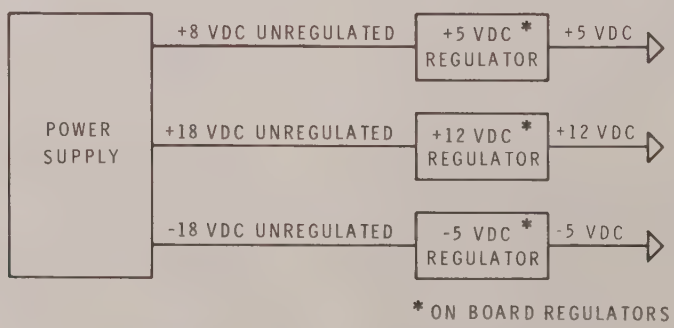
Generating a level 10 interrupt returns the computer to monitor control. Press the ϕ and the RTM/ϕ keys simultaneously generates the level 10 interrupt. The key signals are gated together in IC111A and inverted by IC112. This signal is coupled to the level 10 interrupt line and returned to the microprocessor (IC213).

Pressing the ϕ and the RST/ϕ keys simultaneously generates a hardware reset in IC112, which drives Q119. When Q119 is turned on, the reset line to the CPU is pulled low, causing the CPU to be reinitiated and perform the entire start-up procedure.

POWER SUPPLY

Refer to the Power Supply Block Diagram while you read this Theory of Operation.

The H8 power supply consists of three unregulated supplies. These supplies are +8VDC at 10 amps and ±18VDC at 250 milliamps. Each supply is coupled to the bus for use by each card. Each card has its own on-board regulators to supply +5 VDC. All regulators use over-current protection.



POWER SUPPLY BLOCK DIAGRAM

INSTRUCTION SET

A computer, no matter how sophisticated, can only do what it is "told" to do. A computer is told what to do via a series of coded instructions referred to as a **program**. The realm of the programmer is referred to as **software**, in contrast to the **hardware** that comprises the actual computer equipment. A computer's software refers to all of the programs that have been written for that computer.

When a computer is designed, the engineers provide the Central Processing Unit (CPU) with the ability to perform a particular set of operations. The CPU is designed such that a specific operation is performed when the CPU control logic decodes a particular instruction. Consequently, the operations that can be performed by a CPU define the computer's **instruction set**.

Each computer instruction allows the programmer to initiate the performance of a specific operation. All computers implement certain arithmetic operations in their instruction set, such as an instruction to add the contents of two registers. Often logical operations (for example, OR the contents of two registers) and register operate instructions (for example, increment a register) are included in the instruction set. A computer's instruction set also has instructions that move data between registers, between a register and memory, and between a register and an I/O device. Most instruction sets also provide **conditional instructions**. A conditional instruction specifies an operation to be performed only if certain conditions have been met; for example, jump to a particular instruction if the result of the last operation was zero. Conditional instructions provide a program with a decision-making capability.

By logically organizing a sequence of instructions into a coherent program, the programmer can "tell" the computer to perform a very specific and useful function.

The computer, however, can only execute programs whose instructions are in a binary coded form (for example, a series of 1's and 0's), that is called **machine code**. Because it would be extremely cumbersome to program in machine code, programming languages have been developed. There are programs available which convert the programming language instructions into machine code that can be interpreted by the processor.

One type of programming language is **assembly language**. A unique assembly language mnemonic is assigned to each of the computer's instructions. The programmer can write a program (called the **source program**) using these mnemonics and certain operands; the source program is then converted into machine instructions (called the **object code**). Each assembly language instruction is converted into one machine code instruction (1 or more bytes) by an **assembler** program. Assembly languages are usually machine dependent (for example, they are usually able to run on only one type of computer).

THE 8080 INSTRUCTION SET

The 8080 instruction set includes five different types of instructions:

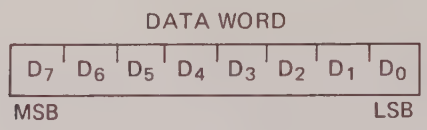
- **Data Transfer Group** — move data between registers or between memory and registers.
- **Arithmetic Group** — add, subtract, increment, or decrement data in registers or in memory.
- **Logical Group** — AND, OR, EXCLUSIVE-OR, compare, rotate, or complement data in registers or in memory.
- **Branch Group** — conditional and unconditional jump instructions, subroutine call instructions, and return instructions.
- **Stack, I/O, and Machine Control Group** — includes I/O instructions, as well as instructions for maintaining the stack and internal control flags.

Instruction and Data Formats

Memory for the 8080 is organized into 8-bit quantities called bytes. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory.

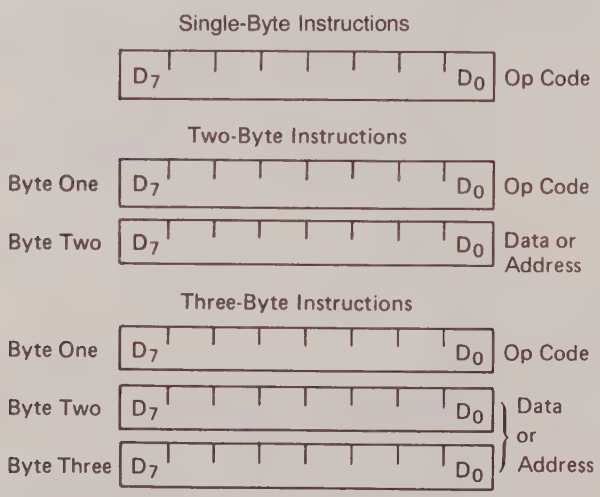
The 8080 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

Data in the 8080 is stored in the form of 8-bit binary integers:



When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the 8080, BIT 0 is referred to as the **Least Significant Bit (LSB)**, and BIT 7 (of an 8-bit number) is referred to as the **Most Significant Bit (MSB).**

The 8080 program instructions may be one, two, or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instructions. The exact instruction format will depend on the particular operation to be executed.


Single-Byte Instructions
D7 D0 Op Code
Two-Byte Instructions
Byte One D7 D0 Op Code
Byte Two D7 D0 Data or Address
Three-Byte Instructions
Byte One D7 D0 Op Code
Byte Two D7 D0 Data or Address
Byte Three D7 D0

Addressing Modes

Often the data to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8080 has four different modes for addressing data stored in memory or in registers:

- **Direct** — Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).

- **Register** — The instruction specifies the register or register pair in which the data is located.

- **Register Indirect** — The instruction specifies a register pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).

- **Immediate** — The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- **Direct** — The branch instruction contains the address of the next instruction to be executed. (Except for the "RST" instruction, byte 2 contains the low-order address and byte 3 the high-order address.)

- **Register Indirect** — The branch instruction indicates a register pair which contains the address of the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special 1-byte call instruction (usually used during interrupt sequences). RST includes a 3-bit field; program control is transferred to the instruction whose address is eight times the contents of this 3-bit field.

Condition Flags

There are five condition flags associated with the execution of instructions on the 8080. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and are each represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1; "reset" by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner.

Zero: If the result of an instruction has the value 0, this flag is set; otherwise it is reset.

Sign: If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.

Parity: If the modulo 2 sum of the bits of the result of the operation is 0 (for example, if the result has even parity), this flag is set; otherwise it is reset (for example, if the result has odd parity).

Carry: If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.

Auxiliary Carry: If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single precision additions, subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

Symbols and Abbreviations

The following symbols and abbreviations are used in the subsequent description of the 8080 instructions:

SYMBOLS	MEANING
accumulator	Register A
addr	16-bit address quantity
data	8-bit data quantity
data 16	16-bit data quantity
byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
port	8-bit address of an I/O device
r, r1, r2	One of the registers A, B, C, D, E, H, L
DDD, SSS	The bit pattern designating one of the registers A, B, C, D, E, H, L (DDD = destination, SSS = source):

DDD or SSS		REGISTER NAME
BINARY	OCTAL	
111	7	A
000	0	B
001	1	C
010	2	D
011	3	E
100	4	H
101	5	L

rp One of the register pairs:

B represents the B, C pair with B as the high-order register and C as the low-order register;

D represents the D, E pair with D as the high-order register and E as the low-order register;

H represents the H, L pair with H as the high-order register and L as the low-order register;

SP represents the 16-bit stack pointer register.

RP The bit pattern designating one of the register pairs B, D, H, SP:

RP	REGISTER PAIR
00	B-C
01	D-E
10	H-L
11	SP

rh The first (high-order) register of a designated register pair.

rl The second (low-order) register of a designated register pair.

PC 16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8-bits, respectively).

SP 16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8-bits, respectively).

rm	Bit m of the register r (bits are numbered 7 through 0 from left to right).
Z, S, P, CY, AC	The condition flags: Zero, Sign, Parity, Carry, and Auxiliary Carry, respectively.
( )	The contents of the memory location or registers enclosed in the parentheses.
←	"Is transferred to"
∧	Logical AND
⊻	Exclusive OR
∨	Inclusive OR
+	Addition
−	Two's complement subtraction
*	Multiplication
↔	"Is exchanged with"
—	The one's complement (e. g., $(\overline{A})$)
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 through 7 respectively.

Description Format

The following pages provide a detailed description of the instruction set of the 8080. Each instruction is described in the following manner:

1. The numbers above the mnemonic are the octal opcodes for the instruction.

2. The assembler format, consisting of the instruction mnemonic and operand fields, is printed in **BOLDFACE** on the left side of the first line.

3. The name of the instruction is enclosed in parentheses on the right side of the first line.

4. The next line(s) contain a symbolic description of the operation of the instruction.

5. This is followed by a narrative description of the operation of the instruction.

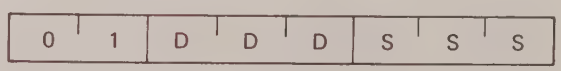
6. The following line(s) contain the binary fields and patterns that comprise the machine instruction.

7. The last four lines contain incidental information about the execution of the instruction. The number of machine cycles and states required to execute the instruction are listed first. If the instruction has two possible execution times, as in a conditional jump, both times will be listed, separated by a slash. Next, any significant data addressing modes (see Page 62) are listed. The last line lists any of the five Flags that are affected by the execution of the instruction.

Data Transfer Group

This group of instructions transfers data to and from registers and memory. **Condition flags are not affected** by any instruction in this group.

1 (0-5,7) (0-5,7)
MOV r1, r2 (Move Register)
(r1) ← (r2)
The content of register r2 is moved to register r1.

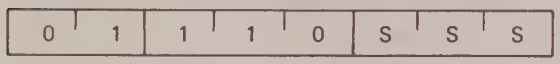


Cycles: 1
States: 5
Addressing: register
Flags: none

1(0-7)6
MOV r, M (Move from memory)
(r) ← ((H) (L))
The content of the memory location, whose address is in registers H and L, is moved to register r.

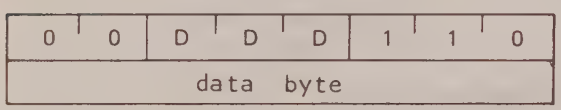
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

16 (0-7)
MOV M, r (Move to memory)
((H) (L)) ← (r)
The content of register r is moved to the memory location whose address is in registers H and L.



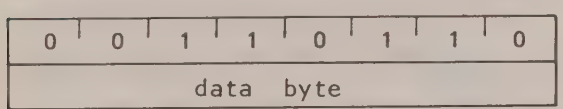
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

0 (0-7)6
MVI r, data (Move Immediate) 0(0-7)6
(r) ← (byte 2)
The content of byte 2 of the instruction is moved to register r.



Cycles: 2
States: 7
Addressing: immediate
Flags: none

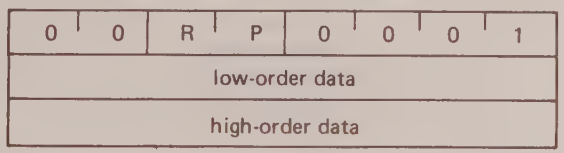
066
MVI M, data (Move to memory immediate)
((H) (L)) ← (byte 2)
The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.



Cycles: 3
States: 10
Addressing: immed./reg. indirect
Flags: none

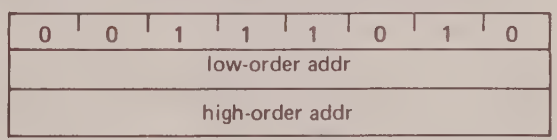
001 (B, C) 041 (H, L)
021 (D, E) 061 (S, P)

LXI rp, data 16 (Load register pair immediate)
(rh) ← (byte 3),
(rl) ← (byte 2)
Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.



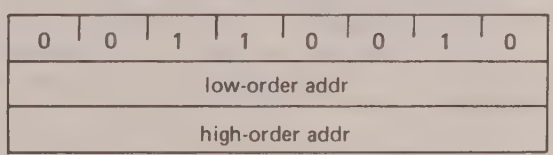
Cycles: 3
States: 10
Addressing: immediate
Flags: none

072
LDA addr (Load Accumulator direct)
(A) ← ((byte 3) (byte 2))
The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to the accumulator.



Cycles: 4
States: 13
Addressing: direct
Flags: none

062
STA addr (Store Accumulator direct)
((byte 3) (byte 2)) ← (A)
The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.



Cycles: 4
States: 13
Addressing: direct
Flags: none

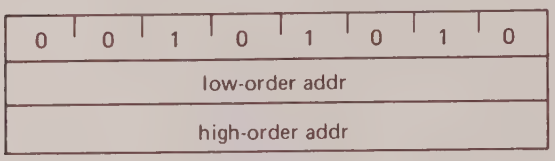
052

LHLD addr (Load H and L direct)

(L) ← ((byte 3) (byte 2))

(H) ← ((byte 3) (byte 2) + 1)

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.



Cycles: 5
States: 16
Addressing: direct
Flags: none

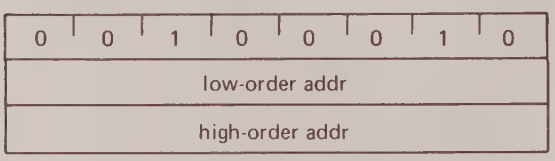
042

SHLD addr (Store H and L direct)

((byte 3) (byte 2)) ← (L)

((byte 3) (byte 2) + 1) ← (H)

The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.



Cycles: 5
States: 16
Addressing: direct
Flags: none

012 (B, C) 032 (D, E)

LDAX rp (Load accumulator indirect)

(A) ← ((rp))

The content of the memory location, whose address is in the register pair rp, is moved to register A. NOTE: Only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

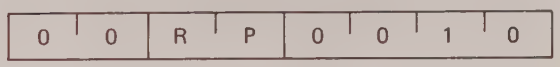
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

002 (B, C) 022 (D, E)

STAX rp (Store accumulator indirect)

((rp)) ← (A)

The content of register A is moved to the memory location whose address is in the register pair rp. NOTE: Only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

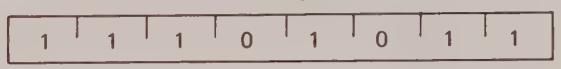
353

XCHG (Exchange H and L with D and E)

(H) ↔(D)

(L) ↔(E)

The contents of registers H and L are exchanged with the contents of registers D and E.



Cycles: 1
States: 4
Addressing: register
Flags: none

Arithmetic Group

This group of instructions performs arithmetic operations on data in registers and memory.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.

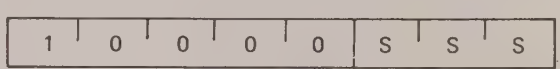
All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

20 (0-5,7)

ADD r (Add Register)

(A) ← (A) + (r)

The content of register r is added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

064

INR M (Increment memory)

((H) (L)) ← ((H) (L)) + 1

The content of the memory location whose address is contained in the H and L registers is incremented by one. NOTE: All condition flags **except CY** are affected.

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

0 (0-5,7)5

DCR r (Decrement Register)

(r) ← (r) − 1

The content of register r is decremented by one. NOTE: All condition flags **except CY** are affected.

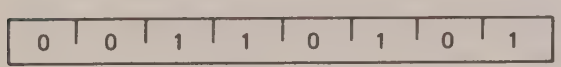
Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

065

DCR M (Decrement memory)

((H) (L)) ← ((H) (L)) − 1

The content of the memory location whose address is contained in the H and L registers is decremented by one. NOTE: All condition flags **except CY** are affected.



Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

003 (B,C) 043 (H,L)
023 (D,E) 063 (S,P)

INX rp (Increment register pair)

(rh) (rl) ← (rh) (rl) + 1

The content of the register pair rp is incremented by one. NOTE: **No condition flags are affected.**

Cycles: 1
States: 5
Addressing: register
Flags: none

013 (B,C) 053 (H,L)
033 (D,E) 073 (S,P)

DCX rp (Decrement register pair)

(rh) (rl) ← (rh) (rl) − 1

The content of the register pair rp is decremented by one. NOTE: **No condition flags are affected.**

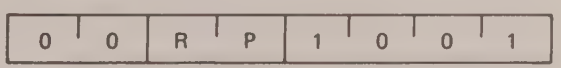
Cycles: 1
States: 5
Addressing: register
Flags: none

011 (B,C) 051 (H,L)
031 (D,E) 071 (S,P)

DAD rp (Add register pair to H and L)

(H) (L) ← (H) (L) + (rh) (rl)

The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. NOTE: **Only the CY flag is affected**. It is set if there is a carry out of the double precision add; otherwise it is reset.



Cycles: 3
States: 10
Addressing: register
Flags: CY

047
DAA (Decimal Adjust Accumulator)

The eight-bit number in the accumulator is adjusted to form two 4-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the least significant 4 bits of the accumulator is greater than 9, **or** if the AC flag is set, 6 is added to the accumulator.

2. If the value of the most significant 4 bits of the accumulator is now greater than 9, **or** if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.

Cycles: 1
States: 4
Flags: Z,S,P,CY,AC

Logical Group:

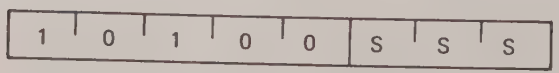
This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

24 (0-5,7)
ANA r (AND Register)

(A) ← (A) ∧ (r)

The content of register r is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared.**

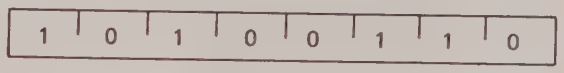


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

246
ANA M (AND memory)

(A) ← (A) ∧ ((H) (L))

The contents of the memory location whose address is contained in the H and L registers is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared.**

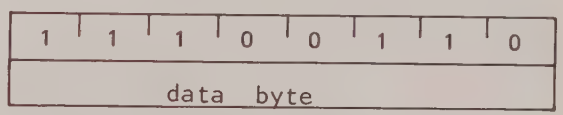


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

346
ANI data (AND immediate)

(A) ← (A) ∧ (byte 2)

The content of the second byte of the instruction is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

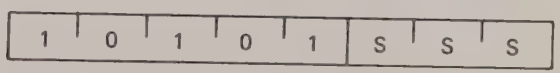


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

25 (0-5,7)
XRA r (Exclusive OR Register)

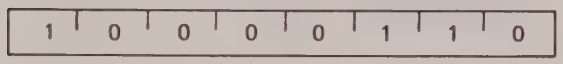
(A) ← (A) ∀ (r)

The content of register r is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

206
ADD M (Add memory)
(A) ← (A) + ((H) (L))
The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.

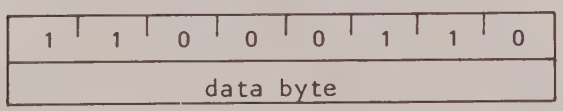


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

216
ADC M (Add memory with carry)
(A) ← (A) + ((H) (L)) + (CY)
The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the content of the accumulator. The result is placed in the accumulator.

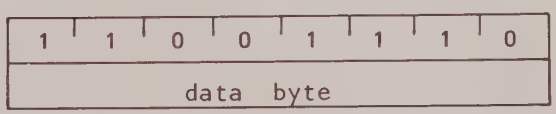
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

306
ADI DATA (Add immediate)
(A) ← (A) + (byte 2)
The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.



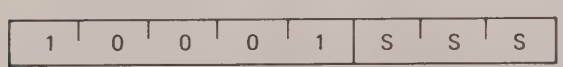
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

316
ACI data (Add immediate with carry)
(A) ← (A) + (byte 2) + (CY)
The content of the second byte of the instruction and the content of the CY flag are added to the content of the accumulator. The result is placed in the accumulator.



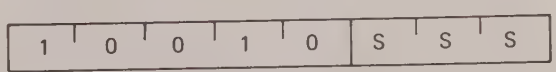
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

21 (0-5,7)
ADC r (Add Register with carry)
(A) ← (A) + (r) + (CY)
The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

22 (0-5,7)
SUB r (Subtract Register)
(A) ← (A) − (r)
The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

226
SUB M (Subtract memory)

(A) ← (A) − ((H) (L))

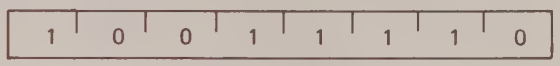
The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

236
SBB M (Subtract memory with borrow)

(A) ← (A) − ((H) (L)) − (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the content of the accumulator. The result is placed in the accumulator.

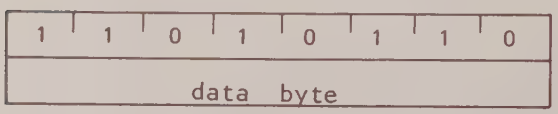


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

326
SUI data (Subtract immediate)

(A) ← (A) − (byte 2)

The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.

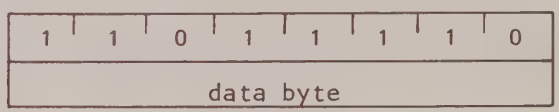


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

336
SBI data (Subtract immediate with borrow)

(A) ← (A) − (byte 2) − (CY)

The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the content of the accumulator. The result is placed in the accumulator.

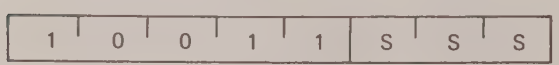


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

23 (0-5,7)
SBB r (Subtract Register with borrow)

(A) ← (A) − (r) − (CY)

The content of register r and the content of the CY flag are both subtracted from the content of the accumulator. The result is placed in the accumulator.



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

0 (0-5,7)4
INR r (Increment Register)

(r) ← (r) + 1

The content of register r is incremented by one.
NOTE: All condition flags **except CY** are affected.

Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

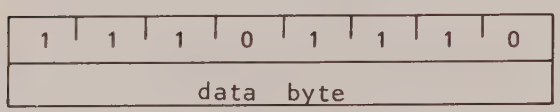
256
XRA M (Exclusive OR Memory)
(A) ← (A) ⊻ ((H) (L))
The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

266
ORA M (OR memory)
(A) ← (A) ∨ ((H) (L))
The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

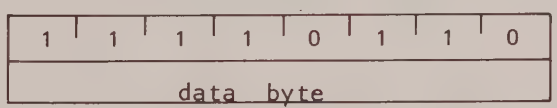
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

356
XRI data (Exclusive OR immediate)
(A) ← (A) ⊻ (byte 2)
The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**



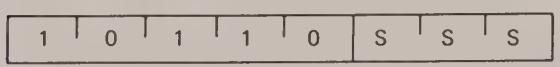
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

366
ORI data (OR Immediate)
(A) ← (A) ∨ (byte 2)
The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**



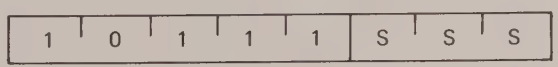
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

26 (0-5,7)
ORA r (OR Register)
(A) ← (A) ∨ (r)
The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

27 (0-5,7)
CMP r (Compare Register)
(A) − (r)
The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = (r). The CY flag is set to 1 if (A) < (r).**



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

276

CMP M (Compare memory)

(A) − ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = ((H) (L)). The CY flag is set to 1 if (A) < ((H) (L)).**

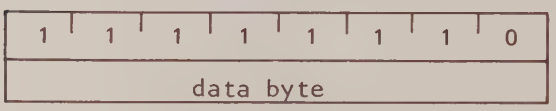
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

376

CPI data (Compare immediate)

(A) − (byte 2)

The content of the second byte of the instruction is subtracted from the content of the accumulator. The condition flags are set by the result of the subtraction. **The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A) <** (byte 2).



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

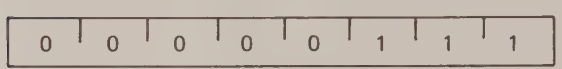
007

RLC (Rotate left)

$(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$

$(CY) \leftarrow (A_7)$

The content of the accumulator is rotated left one position. The low-order bit and the CY flag are both set to the value shifted out of the high-order bit position. **Only the CY flag is affected.**



Cycles: 1
States: 4
Flags: CY

017

RRC (Rotate right)

$(A_n) \leftarrow (A_{n-1})$; $(A_7) \leftarrow (A_0)$

$(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high-order bit and the CY flag are both set to the value shifted out of the low-order bit position. **Only the CY flag is affected.**

Cycles: 1
States: 4
Flags: CY

027

RAL (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n)$; $(CY) \leftarrow (A_7)$

$(A_0) \leftarrow (CY)$

The content of the accumulator is rotated left one position through the CY flag. The low-order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high-order bit. **Only the CY flag is affected.**

Cycles: 1
States: 4
Flags: CY

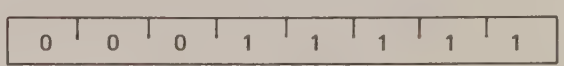
037

RAR (Rotate right through carry)

$(A_n) \leftarrow (A_{n+1})$; $(CY) \leftarrow (A_0)$

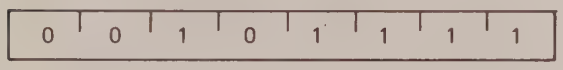
$(A_7) \leftarrow (CY)$

The content of the accumulator is rotated right one position through the CY flag. The high-order bit is set to the CY flag and the CY flag is set to the value shifted out of the low-order bit. **Only the CY flag is affected.**



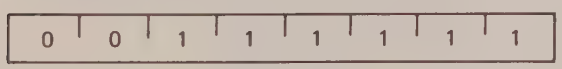
Cycles: 1
States: 4
Flags: CY

057
CMA (Complement accumulator)
(A) ← (A)
The content of the accumulator is complemented (zero bits become 1, one bits become 0). **No flags are affected.**



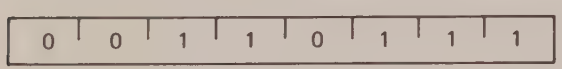
Cycles: 1
States: 4
Flags: none

077
CMC (Complement carry)
(CY) ← ($\overline{CY}$)
The CY flag is complemented. **No other flags are affected.**



Cycles: 1
States: 4
Flags: CY

067
STC (Set carry)
(CY) ← 1
The CY flag is set to 1. **No other flags are affected.**



Cycles: 1
States: 4
Flags: CY

Branch Group

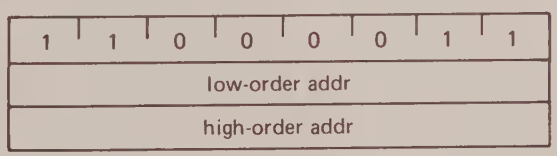
This group of instructions alters normal sequential program flow.

Condition flags are not affected by any instruction in this group.

The two types of branch instructions are **unconditional** and **conditional**. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

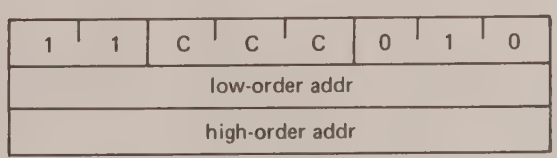
CONDITION	CCC	OCTAL
NZ — not zero (Z = 0)	000	0
Z — zero (Z = 1)	001	1
NC — no carry (CY = 0)	010	2
C — carry (CY = 1)	011	3
PO — parity odd (P = 0)	100	4
PE — parity even (P = 1)	101	5
P — plus (S = 0)	110	6
M — minus (S = 1)	111	7

303
JMP addr (Jump)
(PC) ← (byte 3) (byte 2)
Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



Cycles: 3
States: 10
Addressing: immediate
Flags: none

3 (0-7)2
Jcondition addr (Condition jump)
If (CCC),
(PC) ← (byte 3) (byte 2)
If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.



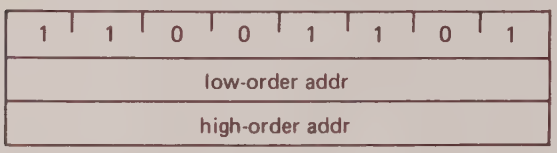
Cycles: 3
States: 10
Addressing: immediate
Flags: none

315

CALL addr (Call)

((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← (byte 3) (byte 2)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



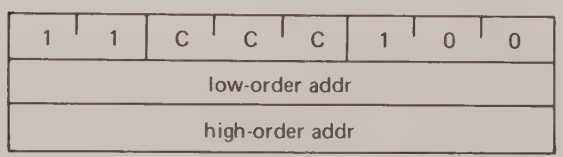
Cycles: 5
States: 17
Addressing: immediate/reg. indirect
Flags: none

3 (0-7) 4

Ccondition addr (Condition call)

If (CCC),
((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← (byte 3) (byte 2)

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.



Cycles: 3/5
States: 11/17
Addressing: immediate/reg. indirect
Flags: none

311

RET (Return)

(PCL) ← ((SP));
(PCH) ← ((SP) + 1);
(SP) ← (SP) + 2;

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.

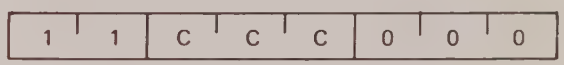
Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

3 (0-7) 0

Rcondition (Conditional return)

If (CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.



Cycles: 1/3
States: 5/11
Addressing: reg. indirect
Flags: none

3 (0-7)7

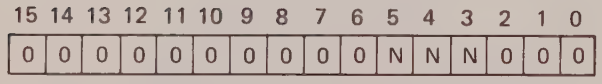
RST n (Restrart)

((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← 8 * (NNN)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction

address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none



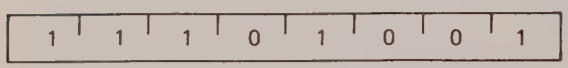
Program Counter After Restart

351

PCHL (Jump H and L indirect — move H and L to PC)

(PCH) ← (H)
(PCL) ← (L)

The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.



Cycles: 1
States: 5
Addressing: register
Flags: none

Stack, I/O, and Machine Control Group

This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, **condition flags are not affected by any instructions in this group.**

305 (B, C) 345 (H, L)
325 (D, E)

PUSH rp (Push)

((SP) − 1) ← (rh)
((SP) − 2) ← (rl)
(SP) ← (SP) − 2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. **NOTE: Register pair rp = SP may not be specified.**

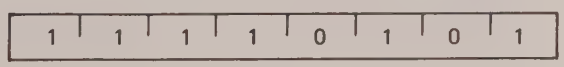
Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

365

PUSH PSW (Push processor status word)

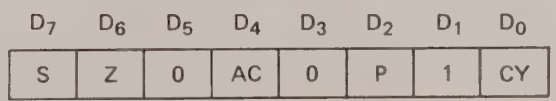
((SP) − 1) ← (A)
((SP) − 2)$_0$ ← (CY), ((SP) − 2)$_1$ ← 1
((SP) − 2)$_2$ ← (P), ((SP) − 2)$_3$ ← 0
((SP) − 2)$_4$ ← (AC), ((SP) − 2)$_5$ ← 0
((SP) − 2)$_6$ ← (Z), ((SP) − 2)$_7$ ← (S)
(SP) ← (SP) − 2

The content of the accumulator is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2.



Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

FLAG WORD



301 (B,C)
321 (D,E)
341 (H,L)

POP rp (Pop)

(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location, whose address is specified by the content of register SP is moved to the low-order register of register pair rp. The content of the memory location whose address is one more than the content of register SP is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. **NOTE: Register pair rp = SP may not be specified.**

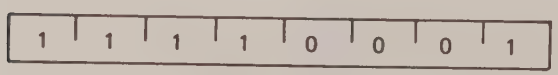
Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

361

POP PSW (Pop processor status word)

$(CY) \leftarrow ((SP))_0$
$(P) \leftarrow ((SP))_2$
$(AC) \leftarrow ((SP))_4$
$(Z) \leftarrow ((SP))_6$
$(S) \leftarrow ((SP))_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.



Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

343

XTHL (Exchange stack top with H and L)

(L) ↔ ((SP))
(H) ↔ ((SP) + 1)

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

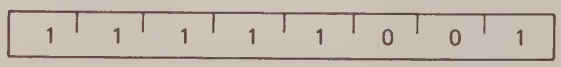
Cycles: 5
States: 18
Addressing: reg. indirect
Flags: none

371

SPHL (Move HL to SP)

(SP) ← (H) (L)

The contents of registers H and L (16 bits) are moved to register SP.



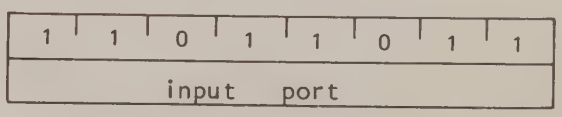
Cycles: 1
States: 5
Addressing: register
Flags: none

333

IN port (Input)

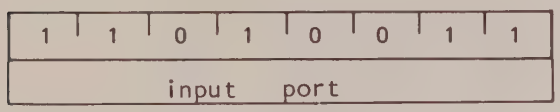
(A)←(data)

The data placed on the eight-bit bi-directional data bus by the specified port is moved to the accumulator.



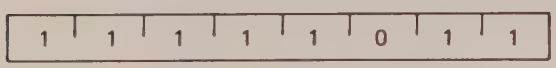
Cycles: 3
States: 10
Addressing: direct
Flags: none

323
OUT port (Output)
(data) ← (A)
The content of the accumulator is placed on the eight-bit bi-directional data bus for transmission to the specified port.



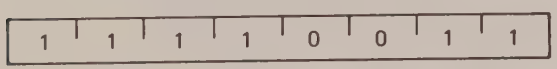
Cycles: 3
States: 10
Addressing: direct
Flags: none

373
EI (Enable interrupt)
The interrupt system is enabled **following the execution of the next instruction.**



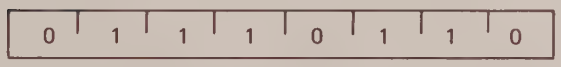
Cycles: 1
States: 4
Flags: none

363
DI (Disable interrupt)
The interrupt system is disabled **immediately following the execution of the DI instruction.**



Cycles: 1
States: 4
Flags: none

166
HLT (Halt)
The processor is stopped. The registers and flags are unaffected.



Cycles: 1
States: 7
Flags: none

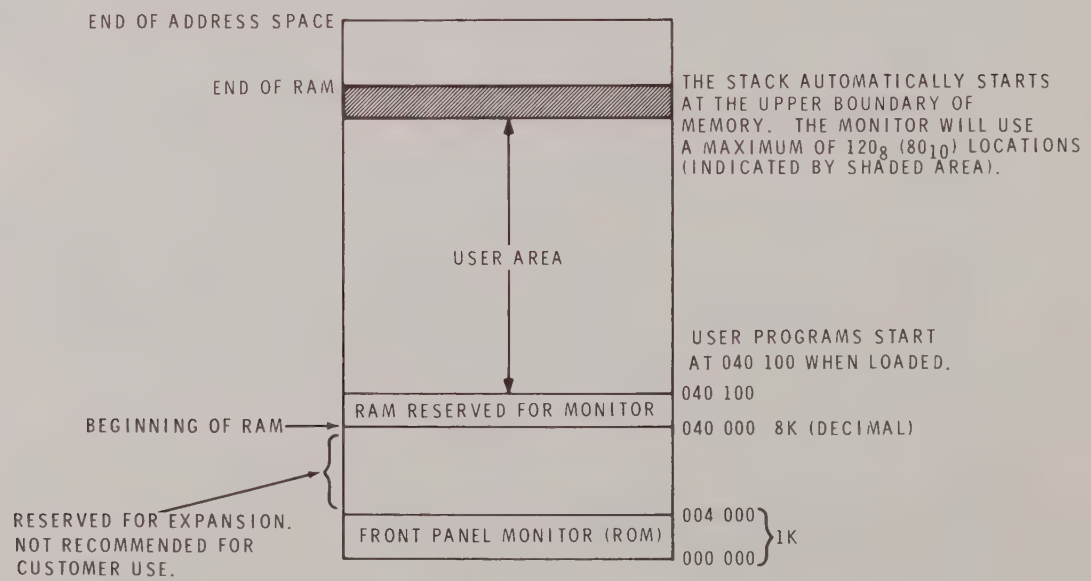
000
NOP (No op)
No operation is performed. The registers and flags are unaffected.

Cycles: 1
States: 4
Flags: none

SYSTEM CONSIDERATIONS

MEMORY MAP

The Memory Map illustrates the use of the specified memory locations.


END OF ADDRESS SPACE
END OF RAM
THE STACK AUTOMATICALLY STARTS AT THE UPPER BOUNDARY OF MEMORY. THE MONITOR WILL USE A MAXIMUM OF 120_8 (80_{10}) LOCATIONS (INDICATED BY SHADED AREA).
USER AREA
USER PROGRAMS START AT 040 100 WHEN LOADED.
040 100
RAM RESERVED FOR MONITOR
BEGINNING OF RAM
040 000 8K (DECIMAL)
RESERVED FOR EXPANSION. NOT RECOMMENDED FOR CUSTOMER USE.
FRONT PANEL MONITOR (ROM)
004 000
000 000
1K

I/O PORT MAP

The I/O Port Map illustrates the use of the specified I/O port addresses.

PORT ADDRESS	
377_8	RESERVED
373_8	CONSOLE CONTROL PORT
372_8	CONSOLE DATA PORT
371_8	LOAD AND DUMP CONTROL PORT
370_8	LOAD AND DUMP DATA PORT
	RESERVED
361_8	FRONT PANEL SEGMENT SELECT
360_8	FRONT PANEL COMMANDS, DIGIT SELECT, AND KEYPAD
300_8	RESERVED

S201-

24	GND*
23	MEMW
22	$\overline{\phi 2}$
21	I/OW
20	RDYIN*
19	M1
18	GND*
17	$\overline{D_7}$
16	$\overline{D_6}$
15	$\overline{D_5}$
14	$\overline{D_4}$
13	$\overline{D_3}$
12	$\overline{D_2}$
11	$\overline{D_1}$
10	$\overline{D_0}$
9	$\overline{INT_2}$*
8	$\overline{INT_1}$*
7	$\overline{INT_7}$
6	$\overline{INT_6}$
5	$\overline{INT_5}$
4	$\overline{INT_4}$
3	$\overline{INT_3}$
2	-18V
1	GND
0	GND

S201-

49	+8V
48	+8V
47	+18V
46	$\overline{\text{ROM DISABLE}}$
45	$\overline{A_{15}}$
44	$\overline{A_{14}}$
43	$\overline{A_{13}}$
42	$\overline{A_{12}}$
41	$\overline{A_{11}}$
40	$\overline{A_{10}}$
39	$\overline{A_9}$
38	$\overline{A_8}$
37	$\overline{A_7}$
36	$\overline{A_6}$
35	$\overline{A_5}$
34	$\overline{A_4}$
33	$\overline{A_3}$
32	$\overline{A_2}$
31	$\overline{A_1}$
30	$\overline{A_0}$
29	$\overline{\text{RESET}}$
28	MEMR
27	HOLD*
26	I/OR
25	HLDA*

*HEATH COMPANY RESERVES THE RIGHT TO CHANGE THESE PIN DESIGNATIONS.

BUS FUNCTIONAL PIN DEFINITIONS

$\overline{A_{15}}$-$\overline{A_0}$ (output, three-state) ADDRESS BUS — The address bus provides the address to memory (up to 64K 8-bit words) or denotes the I/O device number for up to 256 input and 256 output devices. A_0 is the least significant address bit.

$\overline{D_7}$-$\overline{D_0}$ (input/output, thru-state) DATA BUS — The data bus provides bi-directional communication between the CPU, memory, and I/O devices for instructions and data transfers. Also, during the first clock cycle of each machine cycle, the 8080A outputs a status word on the data bus that describes the current machine cycle. $\overline{D_0}$ is the least significant data bit.

$\overline{\phi 2}$ (output) — A CPU board supplied clock phase (TTL compatible).

HOLD (input) — The HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080A address and data bus as soon as the 8080A has completed its use of these buses for the current machine cycle. It is recognized under the following conditions:

1. The CPU is in the HALT state.
2. The CPU is in the T_2 or T_W state and the READY signal is active.

As a result of entering the HOLD state, the CPU address bus and data bus will be in their high impedance state. The CPU acknowledges this state with the HOLD ACKNOWLEDGE (HLDA) pin.

HLDA (output) HOLD ACKNOWLEDGE — The HLDA signal appears in response to the HOLD signal and indicates that the data and address bus will go to their high impedance state. The HLDA signal begins at:

1. T_3 for READ memory or input operation.
2. The clock period following T_3 for WRITE memory or output operation.

In either case, the HLDA signal appears after the rising edge of $\phi 1$ and high impedance occurs after the rising edge of $\phi 2$.

$\overline{INT}_1$-$\overline{INT}_7$ (input) INTERRUPT REQUEST — The CPU recognizes an interrupt request on these lines at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the interrupt enable flip-flop is reset, it will not honor the request.

$\overline{RESET}$ (input) — While the RESET signal is activated, the contents of the program counter are cleared. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip-flops are also reset. The flags, accumulator, stack pointer, and registers are not cleared. NOTE: The RESET signal must be active for a minimum of three clock cycles.

MEMR and I/OR (output) MEMORY READ and INPUT/OUTPUT READ — The read control signals are derived from the logical combination of the appropriate Status Bit (or bits) and the DBIN input from the 8080A.

MEMW and I/OW (output) MEMORY WRITE and INPUT/OUTPUT WRITE — The write control signals are derived from the logical combination of the appropriate Status Bit (or bits) and the $\overline{WR}$ input from the 8080A.

RDYIN (input) READY INPUT — Provides an asynchronous wait request to the clock generator which generates the synchronous READY signal for the microprocessor.

M_1 (output) The first MACHINE CYCLE — Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instruction.

ROM DISABLE (input) ROM DISABLE — An external provision to disable the on-board ROM.

SYSTEM CONFIGURATIONS

Your H8 Digital Computer consists of the central processing unit, front panel, power supply, and motherboard. The mother circuit board is used for installing memory and serial and parallel I/O cards. Memory must be added for the Computer to be operational. Memory size may range from 4K bytes to 32K bytes. An on-board ROM allows the following commands to be executed upon power-up.

1. Memory display and alter.
2. Register display and alter.
3. Input and output to and from a port.
4. Load and dump (with and I/O card and storage device).

Therefore, the Computer can execute machine language programs using the front panel as an I/O device.

You can add the following Heath accessories to expand your Computer system:

Heath Memory Card — Contains 8K of static RAM. A maximum of four Memory Cards (32K total memory) can be installed.

Heath Parallel I/O Interface — Provides three ports of parallel I/O. Each port has eight bits input and eight bits output, with complete hand-shaking available.

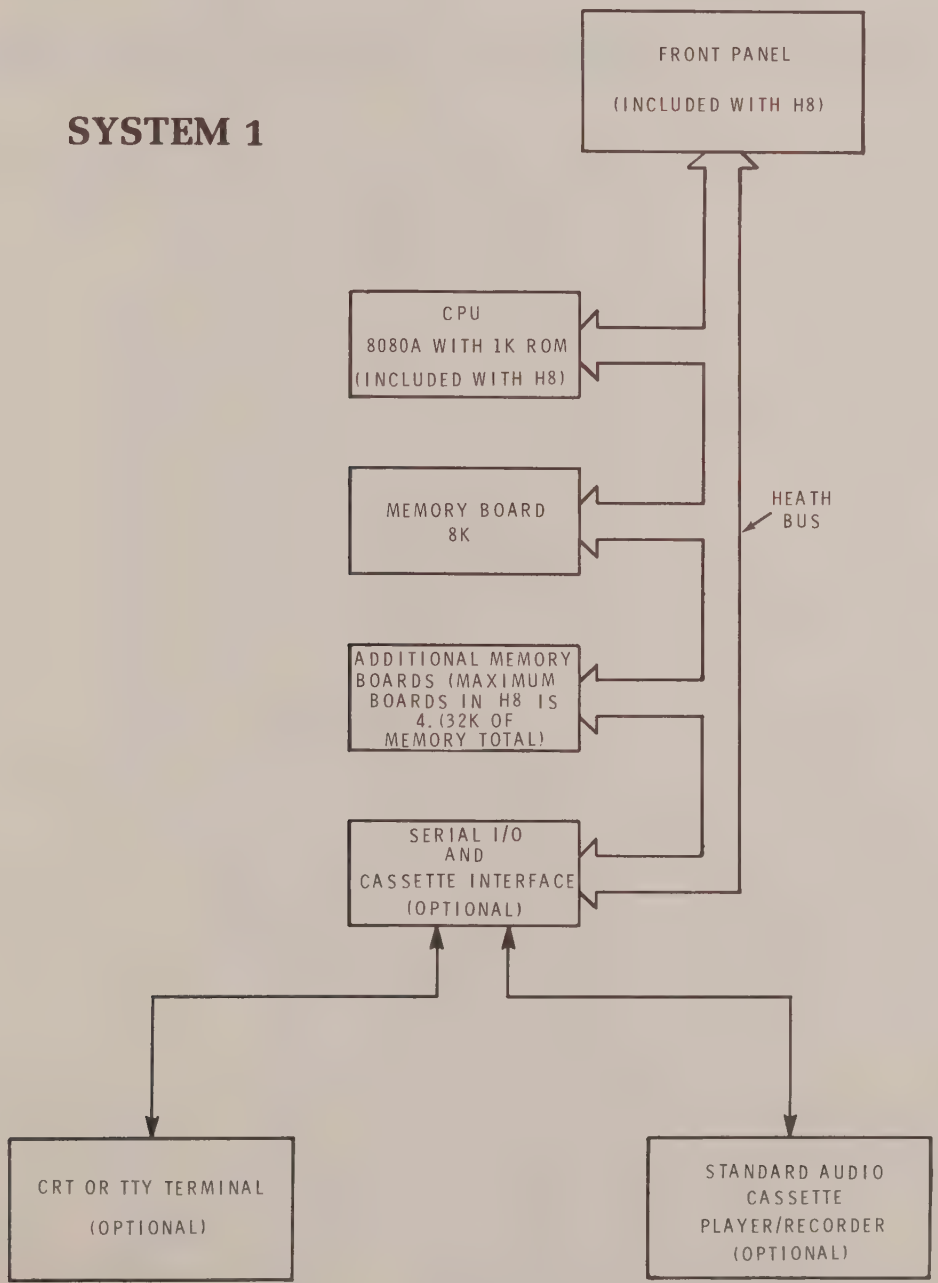
Heath Serial I/O and Cassette Interface — The Cassette Interface allows programs to be stored (dumped) onto tapes and then retrieved (loaded) in the future. The Serial Interface is used with a terminal as a system console or an I/O device.

Heath CRT Terminal — Communicates with the Computer serially. The CRT Terminal can also communicate with the Paper Tape Punch/Reader in parallel.

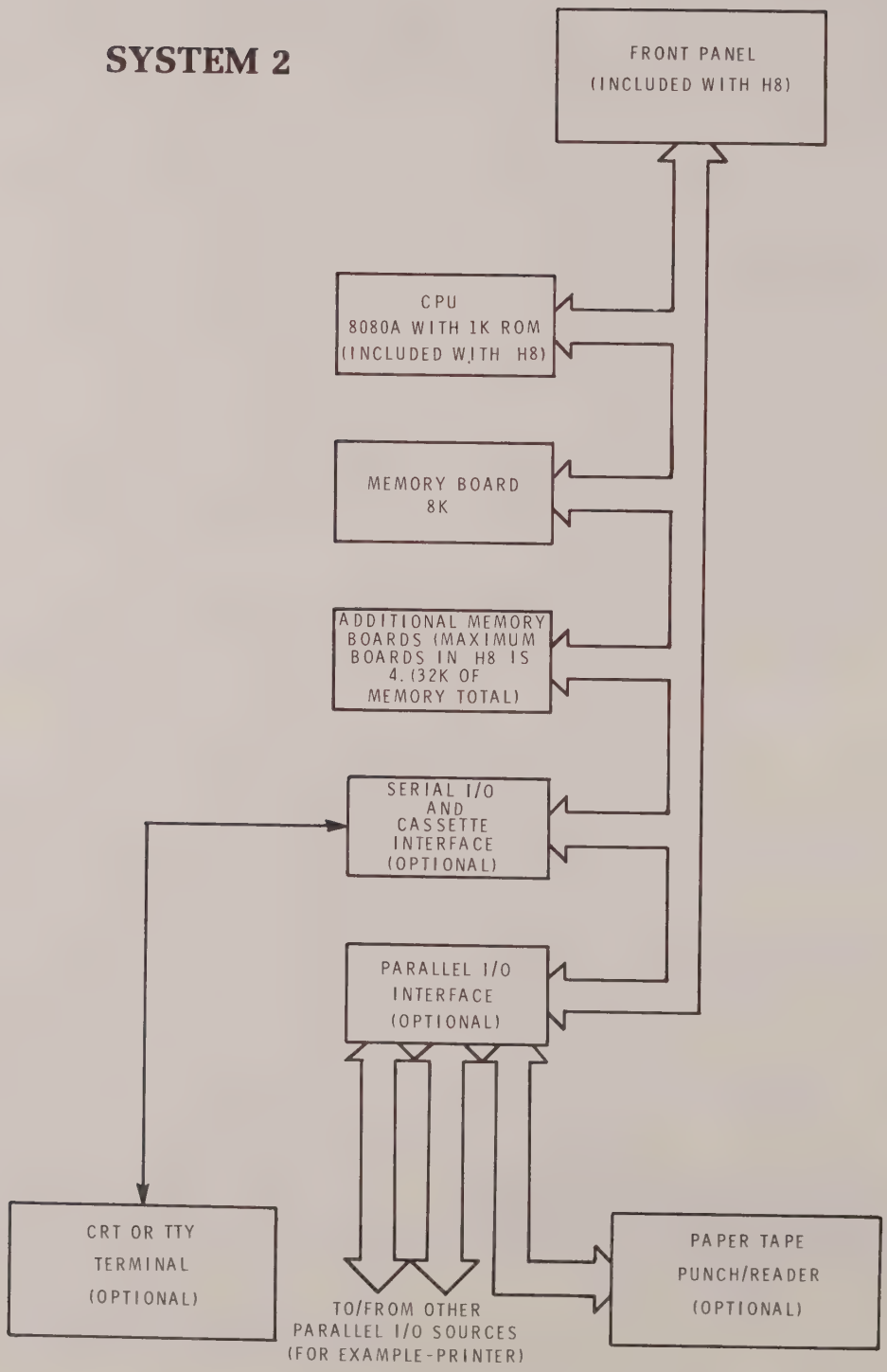
Heath Paper Tape Punch/Reader — Communicates through an 8-bit parallel port.

The following pages show various system configurations available with the H-8 Digital Computer.

SYSTEM 1



SYSTEM 2


FRONT PANEL
(INCLUDED WITH H8)
CPU
8080A WITH 1K ROM
(INCLUDED WITH H8)
MEMORY BOARD
8K
ADDITIONAL MEMORY
BOARDS (MAXIMUM
BOARDS IN H8 IS
4. (32K OF
MEMORY TOTAL)
SERIAL I/O
AND
CASSETTE
INTERFACE
(OPTIONAL)
PARALLEL I/O
INTERFACE
(OPTIONAL)
CRT OR TTY
TERMINAL
(OPTIONAL)
TO/FROM OTHER
PARALLEL I/O SOURCES
(FOR EXAMPLE-PRINTER)
PAPER TAPE
PUNCH/READER
(OPTIONAL)

APPENDIX

SOURCE PROGRAM FOR THE MEMORY TEST ROUTINE

```
* HEATH MEMORY TEST ROUTINE.                                  CAXXXX   02/15/77. 13.12.46.     PAGE      1
  COPYRIGHT 1977, HEATH CO.

                                3  ***     HEATH MEMORY TEST ROUTINE.
                                4  *
                                5  *       COPYRIGHT HEATH COMPANY
                                6  *             BENTON HARBOR, MI. 2/1977
                                7
                                8
                                9
                               10  **      THIS ROUTINE IS USED TO EXERCIZE H8 SYSTEM MEMORY BOARDS.
                               11  *       THE ROUTINE IS ENTERED INTO MEMEORY VIA THE FRONT PANEL, THE
                               12  *       STARTING AND ENDING ADDRESSES ARE ENTERED, AND EXECUTION IS BEGUN.
                               13  *
                               14  *       THE PROGRAM WILL RUN INDEFINITELY IF NO ERRORS ARE FOUND, IT HALTS
                               15  *       IF AN ERROR IS DETECTED.
                               16
                               17
                               18  **      STARTING AND ENDING ADDRESSES:
                               19  *
                               20  *       THE STARTING ADDRESS IS ENTERED (LOW BYTE FIRST) IN LOCATIONS 40101
                               21  *       AND 40102. THE ENDING ADDRESS IS ENTERED (LOW BYTE FIRST) IN LOCATIONS
                               22  *       40104 AND 40105.
                               23
                               24
                               25  **      ERROR HALTS.
                               26  *
                               27  *       THE PROGRAM HALTS WITH 40134 IN THE PC REGISTER IF AN ERROR IS FOUND.
                               28  *       THE HL REGISTER PAIR CONTAINS THE ADDRESS OF THE BYTE IN ERROR.
                               29
                               30
40160                          31  START   EQU   40160A          DEFAULT START ADDRESS
57260                          32  END     EQU   60000A-120Q     DEFAULT END ADDRESS
                               33
40100                          34          ORG   40100A          START OF USER RAM
                               35
                               36
                               37
40100  041 160 040             38  ENTRY   LXI   H,START        (HL) = STARTING ADDRESS
40103  021 260 057             39          LXI   D,END          (DE) = ENDING ADDRESS
                               40
                               41  *       ZERO TEST AREA.
                               42
40106  066 000                 43  MEM1    MVI   M,0            ZERO A BYTE
40110  315 147 040             44          CALL  CMP            SEE IF AT END
40113  043                     45          INX   H
40114  302 106 040             46          JNE   MEM1           NOT YET AT END
                               47
                               48  *       START DIAGNOSING MEMORY, INCREMENT EACH BYTE IN TURN, AND COMPARE
                               49  *       THAT RESULT TO THE EXPECTED VALUE.
                               50
40117  006 000                 51          MVI   B,0            (B) = EXPECTED VALUE
40121  052 101 040             52  MEM2    LHLD  ENTRY+1        (HL) = START ADDRESS
40124  004                     53          INR   B              NEW PASS: EXPECT NEW VALUE
                               54
40125  064                     55  MEM3    INR   M              INCREMENT LOCATION
40126  176                     56          MOV   A,M            (A) = VALUE
```

```
 HEATH MEMORY TEST ROUTINE.                                  CAXXXX   02/15/77. 13.12.46.      PAGE
  COPYRIGHT 1977, HEATH CO.

40127  270              57            CMP   B               COMPARE TO EXPECTED VALUE
40130  312 135 040      58            JE    MEM4            IS OK
                        59
                        60  *         HAVE ERROR. (HL) = ADDRESS OF BYTE IN ERROR.
                        61
40133  166              62            HLT
40134  000              63            NOP
                        64
40135  315 147 040      65  MEM4      CALL  CMP             SEE IF AT END OF RANGE
40140  043              66            INX   H
40141  302 125 040      67            JNE   MEM3            NOT AT END OF PASS YET
40144  303 121 040      68            JMP   MEM2            AT END OF PASS
                        69
                        70
                        71
                        72
                        73
                        74  **        CMP - COMPARE (DE) TO (HL)
                        75  *
                        76  *         EXIT WITH 'Z' SET IF (HL) = (DE)
                        77
40147  172              78  CMP       MOV   A,D
40150  254              79            XRA   H
40151  300              80            RNE                   NOT EQUAL
40152  173              81            MOV   A,E
40153  255              82            XRA   L
40154  311              83            RET                   RETURN WITH 'Z' SET IF EQUAL
                        84
40155                   85            END
```

*

THE FUNCTIONS OF A COMPUTER

This section of the Manual introduces certain basic computer concepts. It provides background information and definitions which will be useful in later sections of this Manual. Those already familiar with computers may skip this material, at their option.

A TYPICAL COMPUTER SYSTEM

A typical digital computer consists of:

a) A central processor unit (CPU)
b) A memory
c) Input/output (I/O) ports

The memory serves as a place to store **instructions,** the coded information that directs the activities of the CPU, and **data**, the coded information processed by the CPU. A group of logically related instructions stored in memory is referred to as a **program**. The CPU "reads" each instruction from memory in a logically determined sequence, and uses it to initiate processing actions. If the program sequence is coherent and logical, processing the program produces intelligible and useful results.

The memory is also used to store the data to be manipulated, as well as the instructions that direct manipulation. The program must be organized such that the CPU does not read a non-instruction word when it expects to see an instruction. The CPU can rapidly access data stored in memory, but often the memory is not large enough to store the data required for a particular application. This problem can be resolved by providing the computer with one or more **input ports.** The CPU can address these ports and input the data contained there. The addition of input ports enables the computer to receive information from external equipment (such as a paper tape reader or floppy disk) at high rates of speed and in large volumes.

A computer also requires one or more **output ports** that permit the CPU to communicate the result of its processing to the outside world. The output may go to a display, for use by a human operator, to a peripheral device that produces "hard copy," such as a line printer, to a peripheral storage device, such as a floppy disk unit, or the output may constitute process control signals that direct the operations of another system, such as an automated assembly line. Like input ports, output ports are addressable. The input and output ports together permit the processor to communicate with the outside world.

The CPU unifies the system. It controls the functions performed by the other components. The CPU fetches instructions from memory, decodes their binary contents and executes them. It also references memory and I/O ports as necessary in the execution of instructions. In addition, the CPU recognizes and responds to certain external control signals, such as interrupt and wait requests. The functional units within a CPU that enable it to perform these functions are described below.

THE ARCHITECTURE OF A CPU

A typical central processor unit (CPU) consists of the following interconnected functional units:

- Registers
- Arithmetic/Logic Unit (ALU)
- Control Circuitry

Registers are temporary storage units within the CPU. Some registers, such as the program counter and instruction register, have dedicated uses. Other registers, such as the accumulator, are for general-purpose use.

Accumulator

The accumulator usually stores one of the operands to be manipulated by the ALU. A typical instruction might direct the ALU to add the contents of some other register to the contents of the accumulator and store the result in the accumulator itself. In general, the accumulator is both a source (operand) and a destination (result) register.

Often a CPU includes a number of additional general purpose registers used to store operands or intermediate data. The availability of general-purpose registers eliminates the need to "shuffle" intermediate results back and forth between memory and the accumulator, thus improving processing speed and efficiency.

Portions of this section of the Manual are reprinted by permission of Intel Corporation (Copyright 1976).

Program Counter (Jumps, Subroutines and the Stack)

The instructions that make up a program are stored in the system's memory. The central processor references the contents of memory in order to determine what action is appropriate. This means the processor must know which location contains the next instruction.

Each of the locations in memory is numbered to distinguish it from all other locations in memory. The number that identifies a memory location is called its **address.**

The processor maintains a counter that contains the address of the next program instruction. This register is called the **program counter**. The processor updates the program counter by adding "1" to the counter each time it fetches an instruction. Therefore, the program counter is always current (pointing to the next instruction).

The programmer therefore stores his instructions in numerically adjacent addresses, so the lower addresses contain the first instructions to be executed and the higher addresses contain later instructions. The only time the programmer may violate this sequential rule is when an instruction in one section of memory is a **jump** instruction to another section of memory.

A jump instruction contains the address of the instruction which is to follow it. The next instruction may be stored in any memory location, as long as the programmed jump specifies the correct address. During the execution of a jump instruction, the processor replaces the contents of its program counter with the address embodied in the instruction. Thus, the logical continuity of the program is maintained.

A special kind of program jump occurs when the stored program "**calls**" a subroutine. In this kind of jump, the processor is required to "remember" the contents of the program counter at the time the call occurs. This enables the processor to resume execution of the main program when it is finished with the last instruction of the subroutine.

A **subroutine** is a program within a program. Usually it is a general-purpose set of instructions that must be executed repeatedly in the course of a main program. Routines which calculate the square, the sine, or the logarithm of a program variable are good examples of functions often written as subroutines. Other examples are programs designed for inputting or outputting data to a particular peripheral device.

The processor has a special way of handling subroutines, in order to insure an orderly return to the main program. When the processor receives a call instruction, it increments the program counter and stores the counter's contents in a reserved memory area known as the **stack**. The stack thus saves the address of the instruction to be executed after the subroutine is completed. Then the processor loads the address specified in the call into its program counter. The next instruction fetched is therefore the first step of the subroutine.

The last instruction in any subroutine is a **return**. Such an instruction need specify no address. When the processor fetches a return instruction, it simply replaces the current contents of the program counter with the address on the top of the stack. This causes the processor to resume execution of the program at the point immediately following the original call instruction.

Subroutines are often **nested**; that is, one subroutine will sometimes call a second subroutine. The second may call a third, and so on. This is perfectly acceptable, as long as the processor has enough capacity to store the necessary return addresses, and the logical provision for doing so. In other words, the maximum depth of nesting is determined by the depth of the stack itself. If the stack has space for storing three return addresses, then three levels of subroutine nesting may be accommodated.

Processors have different ways of maintaining stacks. Some have facilities for the storage of return addresses built into the processor itself. Other processors use a reserved area of external memory as the stack and simply maintain a **pointer** register which contains the address of the most recent stack entry. The external stack allows virtually unlimited subroutine nesting. In addition, if the processor provides instructions that cause the contents of the accumulator and other general-purpose registers to be "pushed" onto the stack or "popped" off the stack via the address stored in the stack pointer, multi-level interrupt processing (described later in this section) is possible. The status of the processor (for example, the contents of all the registers) can be saved in the stack when an interrupt is accepted and then restored after the interrupt has been serviced. This ability to save the processor's status at any given time is possible even if an interrupt service routine, itself, is interrupted.

Instruction Register and Decoder

Every computer has a **word length** characteristic of that machine. A computer's word length is usually determined by the size of its internal storage elements and interconnecting paths (referred to as **buses**); for example, a computer whose registers and buses can store and transfer eight bits of information has a characteristic word length of eight bits and is referred to as an 8-bit parallel processor. An 8-bit parallel processor generally finds it most efficient to deal with 8-bit binary fields, and the memory associated with such a processor is therefore organized to store eight bits in each addressable memory location. Data and instructions are stored in memory as 8-bit binary numbers, or as numbers that are integral multiples of eight bits: 16 bits, 24 bits, and so on. This characteristic 8-bit field is often referred to as a **byte**.

Each operation the processor can perform is identified by a unique byte of data known as an **instruction code** or **operation code**. An 8-bit word used as an instruction code can distinguish between 256 alternative actions, more than adequate for most processors.

The processor fetches an instruction in two distinct operations. First, the processor transmits the address in its program counter to the memory. Then the memory returns the addressed byte to the processor. The CPU stores this instruction byte in the **instruction register**, and uses it to direct activities during the remainder of the instruction execution.

The mechanism by which the processor translates an instruction code into specific processing actions requires a more elaborate explanation than is given here. The concept, however, should be intuitively clear to any logic designer. The eight bits stored in the instruction register can be decoded and used to selectively activate one of a number of output lines, in this case up to 256 lines. Each line represents a set of activities associated with execution of a particular instruction code. The enabled line can be combined with selected timing pulses to develop electrical signals that can then be used to initiate specific actions. This translation of code into action is performed by the **instruction decoder** and the associated control circuitry.

An 8-bit instruction code is often sufficient to specify a particular processing action. There are times, however, when execution of the instruction requires more information than eight bits can convey.

One example of this is when the instruction references a memory location. The basic instruction code identifies the operation to be performed, but cannot specify the object address as well. In a case like this, a two- or three-byte instruction must be used. Successive instruction bytes are stored in sequentially adjacent memory locations, and the processor performs two or three fetches in succession to obtain the full instruction. The first byte retrieved from memory is placed in the processor's instruction register, and subsequent bytes are placed in temporary storage; the processor then proceeds with the execution phase. Such an instruction is referred to as **variable length**.

Address Register(s)

A CPU may use a register or register pair to hold the address of a memory location to be accessed for data. If the address register is **programmable**, (for example, if there are instructions that allow the programmer to alter the contents of the register) the program can "build" an address in the address register prior to executing a **memory reference** instruction (for example, an instruction that reads data from memory, writes data to memory, or operates on data stored in memory).

Arithmetic/Logic Unit (ALU)

All processors contain an arithmetic/logic unit, often referred to simply as the **ALU**. The ALU, as its name implies, is that portion of the CPU hardware which performs the arithmetic and logical operations on the binary data.

The ALU must contain an **adder** capable of combining the contents of two registers in accordance with the logic of binary arithmetic. This provision permits the processor to perform arithmetic manipulations on the data it obtains from memory and from its other inputs.

Using only the basic adder, a capable programmer can write routines which will subtract, multiply and divide, giving the machine complete arithmetic capabilities. In practice, however, most ALU's provide other built-in functions, including hardware subtraction, Boolean logic operations, and shift capabilities.

The ALU contains **flag bits** which specify certain conditions that arise in arithmetic and logical manipulations. Flags typically include **carry, zero, sign**, and **parity**. It is possible to program jumps which are conditionally dependent on the status of one or more flags. Thus, for example, the program may be designed to jump to a special routine if the carry bit is set following an addition instruction.

Control Circuitry

The control circuitry is the primary functional unit within a CPU. Using clock inputs, the control circuitry maintains the proper sequence of events required for any processing task. After an instruction is fetched and decoded, the control circuitry issues the appropriate signals (to units both internal and external to the CPU) for initiating the proper processing action. Often the control circuitry is capable of responding to external signals, such as an interrupt or wait request. An **interrupt** request causes the control circuitry to temporarily interrupt main program execution, jump to a special routine to service the interrupting device, then automatically return to the main program. A **wait** request is often issued by a memory or I/O element that operates slower than the CPU. The control circuitry will idle the CPU until the memory or I/O port is ready with the data.

COMPUTER OPERATIONS

There are certain operations basic to almost any computer. A sound understanding of these basic operations is a necessary prerequisite to examining the specific operations of a particular computer.

Timing

The activities of the central processor are cyclical. The processor fetches an instruction, performs the operations required, fetches the next instruction, and so on. This orderly sequence of events requires precise timing, and the CPU therefore requires a free-running oscillator clock that furnishes the reference for all processor actions. The combined fetch and execution of a single instruction is referred to as an **instruction cycle**. The portion of a cycle identified with a clearly defined activity is called a **state**. And the interval between pulses of the timing oscillator is referred to as a **clock period**. As a general rule, one or more clock periods are necessary for the completion of a state, and there are several states in a cycle.

Instruction Fetch

The first state(s) of any instruction cycle is dedicated to fetching the next instruction. The CPU issues a read signal and the contents of the program counter are sent to memory, which responds by returning the next instruction word. The first byte of the instruction is placed in the instruction register. If the instruction consists of more than one byte, additional states are required to fetch each byte of the instruction. When the entire instruction is present in the CPU, the program counter is incremented (in preparation for the next instruction fetch) and the instruction is decoded. The operation specified in the instruction will be executed in the remaining states of the instruction cycle. The instruction may call for a memory read or write, an input or output and/or internal CPU operation, such as a register-to-register transfer or an add-registers operation.

Memory Read

An instruction **fetch** is merely a special memory read operation that brings the instruction to the CPU's instruction register. The instruction fetched may then call for data to be read from memory into the CPU. The CPU again issues a read signal and sends the proper memory address; memory responds by returning the requested word. The data received is placed in the accumulator or one of the other general-purpose registers (not the instruction register).

Memory Write

A memory write operation is similar to a read except for the direction of data flow. The CPU issues a write signal, sends the proper memory address, then sends the data word to be written into the addressed memory location.

Wait (Memory Synchronization)

As previously stated, the activities of the processor are timed by a master clock oscillator. The clock period determines the timing of all processing activity.

The speed of the processing cycle is limited by the memory's **access time**. Once the processor has sent a read address to memory, it cannot proceed until the memory has had time to respond. Most memories are capable of responding much faster than the processing cycle requires. A few, however, cannot supply the addressed byte within the minimum time established by the processor's clock.

Therefore, a processor contains a synchronization provision, which permits the memory to request a **wait state**. When the memory receives a read or write enable signal, it places a request signal on the processor's READY line, causing the CPU to idle temporarily. After the memory has had time to respond, it frees the processor's READY line, and the instruction cycle proceeds.

Input/Output

Input and Output operations are similar to memory read and write operations with the exception that a peripheral I/O device is addressed instead of a memory location. The CPU issues the appropriate input or output control signal, sends the proper device address, and either receives the data being input or sends the data to be output.

Data can be input/output in either parallel or serial form. All data within a digital computer is represented in binary coded form. A binary data word consists of a group of bits; each bit is either a one or a zero. **Parallel** I/O consists of transferring all bits in the word at the same time, one bit per line. **Serial** I/O consists of transferring one bit at a time on a single line. Naturally, serial I/O is much slower, but it requires considerably less hardware than does parallel I/O.

Interrupts

Interrupt provisions are included on many central processors as a means of improving the processor's efficiency. Consider the case of a computer processing a large volume of data, portions of which are to be output to a printer. The CPU can output a byte of data within a single machine cycle but it may take the printer the equivalent of many machine cycles to actually print the character specified by the data byte. The CPU could then remain idle, waiting until the printer can accept the next data byte. If an interrupt capability is implemented on the computer, the CPU can output a data byte, then return to data processing. When the printer is ready to accept the next data byte, it can request an interrupt. When the CPU acknowledges the interrupt, it suspends main program execution and automatically branches to a routine that will output the next data byte. After the byte is output, the CPU continues with main program execution. Note that this is, in principle, quite similar to a subroutine call, except the jump is initiated externally rather than by the program.

More complex interrupt structures are possible in which several interrupting devices share the same processor but have different priority levels. Interruptive processing is an important feature that enables maximum utilization of a processor's capacity for high system throughput.

Hold

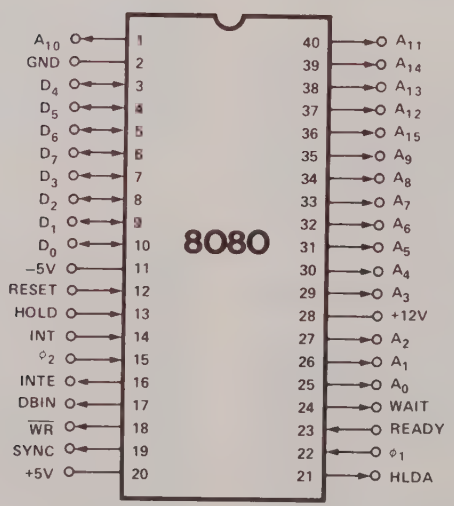
Another important feature that improves the throughput of a processor is the **hold**. The hold provision enables **Direct Memory Access** (DMA) operations.

In ordinary input and output operations, the processor supervises the entire data transfer. Information to be placed in memory is tranferred from the input device to the processor, and then from the processor to the designated memory location. In similar fashion, information that goes from memory to output devices goes by way of the processor.

Some peripheral devices, however, are capable of transferring information to and from memory much faster than the processor itself can accomplish the transfer. If any appreciable quantity of data must be transferred to or from such a device, **system throughput** will be increased by having the device accomplish the transfer directly. The processor must temporarily suspend its operation during such a transfer to prevent conflicts that would arise if processor and peripheral device attempted to access memory simultaneously. Therefore, a **hold** provision is included on some processors.

THE 8080 CENTRAL PROCESSOR UNIT

The 8080 is a complete 8-bit parallel, central processor unit (CPU) for use in general-purpose digital computer systems. It is fabricated on a single LSI chip, using Intel's n-channel silicon gate MOS process. The 8080 transfers data and internal state information via an 8-bit, bi-directional 3-state data bus (D_0-D_7). Memory and peripheral device addresses are transmitted over a separate 16-bit 3-state address bus (A_0-A_{15}). Six timing and control outputs (SYNC, DBIN, WAIT, WR, HLDA and INTE) emanate from the 8080, while four control inputs (READY, HOLD, INT and RESET), four power inputs (+12V, +5V, −5V, and GND), and two clock inputs (ϕ_1 and ϕ_2) are accepted by the 8080.



PICTORIAL 7-1

8080 Pin Designations

ARCHITECTURE OF THE 8080 CPU

The 8080 CPU consists of the following functional units:

- Register array and address logic
- Arithmetic and logic unit (ALU)
- Instruction register and control section
- Bi-directional, 3-state data bus buffer

Pictorial 7-2 illustrates the functional blocks within the 8080 CPU.

Registers

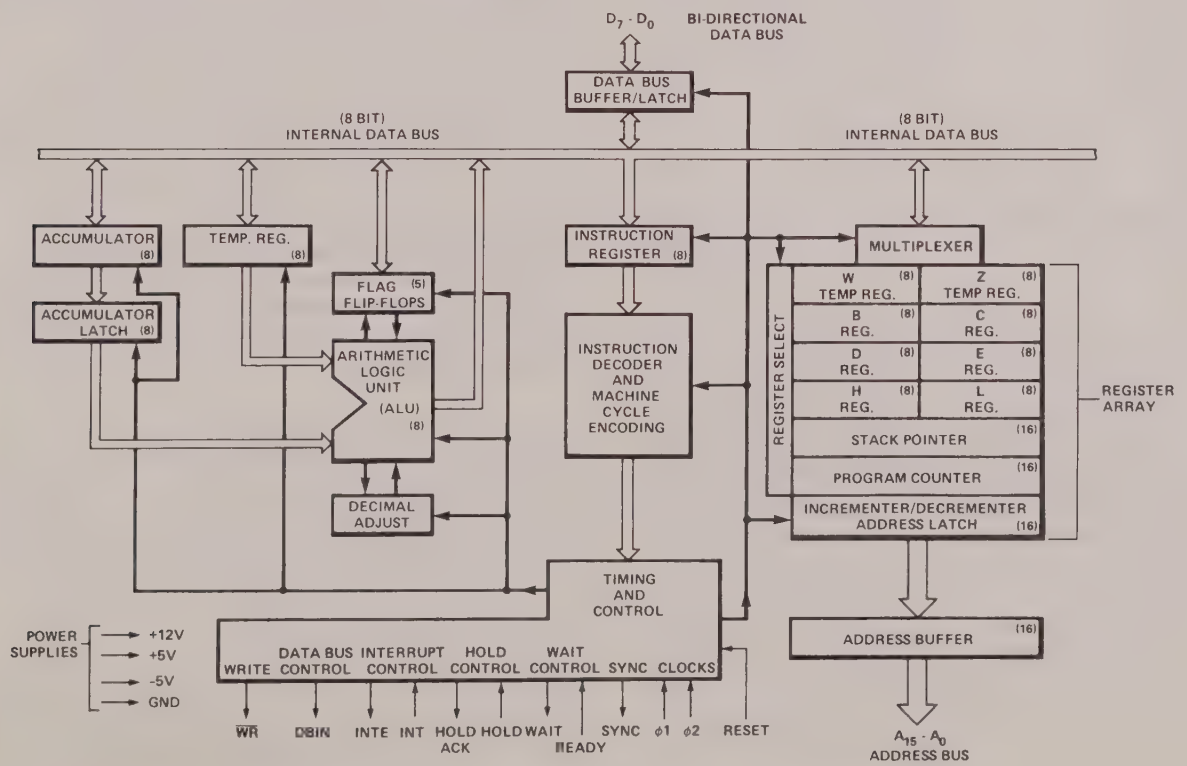
The register section consists of a static RAM array organized into six 16-bit registers.

- Program counter (PC)
- Stack pointer (SP)
- Six 8-bit, general-purpose registers arranged in pairs, referred to as B, C; D, E; and H, L.
- A temporary register pair called W, Z.

The program counter maintains the memory address of the current program instruction and is incremented automatically during every instruction fetch. The stack pointer maintains the address of the next available stack location in memory. The stack pointer can be initialized to use any portion of read-write memory as a stack. The stack pointer is decremented when data is "pushed" onto the stack and incremented when data is "popped" off the stack (for example, the stack grows "downward").

The six general-purpose registers can be used either as single registers (8-bit) or as register pairs (16-bit). The temporary register pair, W, Z, is not program addressable and is only used for the internal execution of instructions.

8-bit data bytes can be transferred between the internal bus and the register array via the register-select multiplexer. 16-bit transfers can proceed between the register array and the address latch or the incrementer/decrementer circuit. The address latch receives data from any of the three register pairs and drives the 16 address output buffers (A_0-A_{15}), as well as the incrementer/decrementer circuit. The incrementer/decrementer circuit receives data from the address latch and sends it to the register array. The 16-bit data can be incremented, decremented, or transferred between registers.



PICTORIAL 7-2 8080 CPU Functional Block Diagram

Arithmetic and Logic Unit (ALU)

The ALU contains the following registers:

- An 8-bit accumulator
- An 8-bit temporary accumulator (ACT)
- A 5-bit flag register: zero, carry, sign, parity and auxiliary carry
- An 8-bit temporary register (TMP)

Arithmetic, logical, and rotate operations are performed in the ALU. The ALU is fed by the temporary register (TMP) and the temporary accumulator (ACT) and carry flip-flop. The result of the operation can be transferred to the internal bus or to the accumulator; the ALU also feeds the flag register.

The temporary register (TMP) receives information from the internal bus and can send all or portions of it to the ALU, the flag register, and the internal bus.

The accumulator (ACC) can be loaded from the ALU and the internal bus and can transfer data to the temporary accumulator (ACT) and the internal bus. The contents of the accumulator (ACC) and the auxiliary carry flip-flop can be tested for decimal correction during the execution of the DAA instruction (see the Instruction Set).

Instruction Register and Control

During an instruction fetch, the first byte of an instruction (containing the OP code) is transferred from the internal bus to the 8-bit instruction register.

The contents of the instruction register are, in turn, available to the instruction decoder. The output of the decoder, combined with various timing signals, provides the control signals for the register array, ALU, and data buffer blocks. In addition, the outputs from the instruction decoder and external control signals feed the timing and state control section which generates the state and cycle timing signals.

Data Bus Buffer

This 8-bit, bi-directional, 3-state buffer is used to isolate the CPU's internal bus from the external data bus (D_0 through D_7). In the output mode, the internal bus content is loaded into an 8-bit latch that, in turn, drives the data bus output buffers. The output buffers are switched off during input or non-transfer operations.

During the input mode, data from the external data bus is transferred to the internal bus. The internal bus is precharged at the beginning of each internal state, except for the transfer state (T_3-described later).

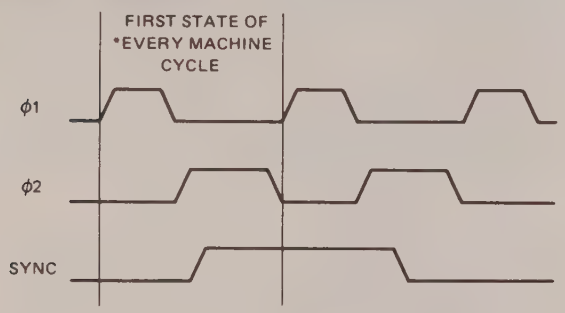
THE PROCESSOR CYCLE

An **instruction cycle** is the time required to fetch and execute an instruction. During the fetch, a selected instruction (one, two or three bytes) is extracted from memory and deposited in the CPU's instruction register. During the execution phase, the instruction is decoded and translated into specific processing activities.

Every instruction cycle consists of one, two, three, four, or five machine cycles. A **machine cycle** is required each time the CPU accesses memory or an I/O port. The fetch portion of an instruction cycle requires one machine cycle for each byte to be fetched. The duration of the execution portion of the instruction cycle depends on the kid of instruction that has been fetched. Some instructions do not require any machine cycles other than those necessary to fetch the instruction; other instructions, however, require additional machine cycles to write or read data to/from memory or I/O devices. The DAD instruction is an exception in that it requires two additional machine cycles to complete an internal register-pair add (see the Instruction Set).

Each machine cycle consists of three, four, or five states. A state is the smallest unit of processing activity and is the interval between two successive positive-going transitiitions of the $\phi 1$ driven clock pulse. The 8080 is driven by a 2-phase clock oscil-

lator. All processing activities are referenced to the period of this clock. The two non-overlapping clock pulses, $\phi 1$ and $\phi 2$, are furnished by external circuitry. The $\phi 1$ clock pulse divides each machine cycle into states. Timing logic within the 8080 uses the clock inputs to produce a SYNC pulse, which identifies the beginning of every machine cycle. The SYNC pulse is triggered by the low-to-high transition of $\phi 2$, as shown in Pictorial 7-3.



*SYNC DOES NOT OCCUR IN THE SECOND AND THIRD MACHINE CYCLES OF A DAD INSTRUCTION SINCE THESE MACHINE CYCLES ARE USED FOR AN INTERNAL REGISTER-PAIR ADD.

PICTORIAL 7-3 $\phi 1$, $\phi 2$, and SYNC Timing

There are three exceptions to the defined duration of a state. They are the WAIT state, the hold (HLDA) state, and the halt (HLTA) state, described later in this section. Because the WAIT, HLDA, and HLTA states depend upon external events, they are of indeterminate length. Even these exceptional states, however, must be synchronized with the pulses of the driving clock. Thus, the duration of all states are integral multiples of the clock period.

To summarize then, each **clock period** marks a **state**; three to five states constitute a machine cycle; and one to five **machine cycles** comprise an **instruction cycle**. A full instruction cycle requires anywhere from four to eighteen states for its completion, depending on the kind of instruction involved.

Machine Cycle Identification

With the exception of the DAD instruction, there is just one consideration that determines how many machine cycles are required in any given instruction cycle: the number of times the processor must reference a memory address or an addressable peripheral device, in order to fetch and execute the instruction. Like many processors, the 8080 is so constructed that it can transmit only one address per machine cycle. Thus, if the fetch and execution of an instruction requires two memory references, then the instruction cycle associated with that instruction consists of two machine cycles. If five such references are called for, the instruction cycle contains five machine cycles.

Every instruction cycle has at least one reference to memory, during which the instruction is fetched. An instruction cycle must always have a fetch, even if the execution of the instruction requires no further references to memory. The first machine cycle in every instruction cycle is therefore a fetch. Beyond that, there are no fast rules. It depends on the kind of instruction that is fetched.

Consider some examples. The add-register (ADD r) instruction is an instruction that requires only a single machine cycle (fetch) for its completion. In this one-byte instruction, the contents of one of the CPU's six general-purpose registers is added to the existing contents of the accumulator. Since all the information necessary to execute the command is contained in the eight bits of the instruction code, only one memory reference is necessary. Three states are used to extract the instruction from memory, and one additional state is used to accomplish the desired addition. The entire instruction cycle, thus, requires only one machine cycle that consists of four states, or four periods of the external clock.

Suppose now, however, that you wish to add the contents of a specific memory location to the existing contents of the accumulator (ADD M). Although this is quite similar in principle to the example just cited, several additional steps will be used. An extra machine cycle will be used in order to address the desired memory location.

The actual sequence is as follows. First the processor extracts from memory the 1-byte instruction word addressed by its program counter. This takes three states. The 8-bit instruction word obtained during the fetch machine cycle is deposited in the CPU's instruction register and used to direct activities during the remainder of the instruction cycle. Next, the processor sends out, as an address, the contents of its H and L registers. The 8-bit data word returned during this MEMORY READ machine cycle is placed in a temporary register inside the 8080. By now, three more clock periods (states) have elapsed. In the seventh and final state, the contents of the temporary register are added to those of the accumulator. Two machine cycles, consisting of several states in all, complete the "ADD M" instruction cycle.

At the opposite extreme is the save H and L registers (SHLD) instruction, which requires five machine cycles. During an "SHLD" instruction cycle, the contents of the processor's H and L registers are deposited in two sequentially adjacent memory locations; the destination is indicated by two address bytes which are stored in the two memory locations immediately following the operation code byte. The following sequence of events occurs:

(1) A fetch machine cycle, consisting of four states. During the first three states of this machine cycle, the processor fetches the instruction indicated by its program counter. The program counter is then incremented. The fourth state is used for internal instruction decoding.

(2) A MEMORY READ machine cycle, consisting of three states. During this machine cycle, the byte indicated by the program counter is read from memory and placed in the processor's Z register. The program counter is incremented again.

(3) Another MEMORY READ machine cycle, consisting of three states, in which the byte indicated by the processor's program counter is read from memory and placed in the W register. The program counter is incremented in anticipation of the next instruction fetch.

(4) A MEMORY WRITE machine cycle, of three states, in which the contents of the L register are transferred to the memory location pointed to by the present contents of the W and Z registers. The state following the transfer is used to increment the W, Z register pair so it indicates the next memory location to receive data.

(5) A MEMORY WRITE machine cycle, of three states, in which the contents of the H register are transferred to the new memory location pointed to by the W, Z register pair.

In summary, the "SHLD" instruction cycle contains five machine cycles and takes 16 states to execute.

Most instructions fall somewhere between the extremes typified by the "ADD r" and the "SHLD" instructions. The input (INP) and the output (OUT) instructions, for example, require three machine cycles: a FETCH, to obtain the instruction; a MEMORY READ, to obtain the address of the object peripheral; and an INPUT or an OUTPUT machine cycle, to complete the transfer.

While no one instruction cycle will consist of more than five machine cycles, the following ten different types of machine cycles may occur within an instruction cycle:

(1) FETCH (M_1)
(2) MEMORY READ
(3) MEMORY WRITE
(4) STACK READ
(5) STACK WRITE
(6) INPUT
(7) OUTPUT
(8) INTERRUPT
(9) HALT
(10) HALT • INTERRUPT

The machine cycles that actually do occur in a particular instruction cycle depend upon the kind of instruction, with the overriding stipulation that the first machine cycle in any instruction cycle is always a fetch.

The processor identifies the machine cycle in progress by transmitting an 8-bit status word during the first state of every machine cycle. Updated status information is presented on the 8080's data lines (D_0-D_7), during the SYNC interval. This data is saved in latches, and used to develop control signals for external circuitry. Table 7-1 shows how the positive-true status information is distributed on the processor's data bus.

Status signals are provided principally for the control of external circuitry. Simplicity of interface, rather than machine cycle identification, dictates the logical definition of individual status bits. You will therefore observe that certain processor machine cycles are uniquely identified by a single status bit, but others are not. The M_1 status bit (D_6), for example, unambiguously identifies a fetch machine cycle. A stack read, on the other hand, is indicated by the coincidence of STACK and MEMR signals. Machine cycle identification data is also valuable in the test and de-bugging phases of system development. Table 7-1 lists the status bit outputs for each type of machine cycle.

State Transition Sequence

Every machine cycle within an instruction cycle consists of three to five active states (referred to as T_1, T_2, T_3, T_4, T_5 or T_W). The actual number of states depends upon the instruction being executed, and on the particular machine cycle within the greater instruction cycle. The state transition diagram in Pictorial 7-4 shows how the 8080 proceeds from state to state in the course of a machine cycle. The diagram also shows how the ready, hold, and interrupt lines are sampled during the machine cycle, and how the conditions on these lines may modify the basic transition sequence. In the present discussion, we are concerned only with the basic sequence and with the ready function. The hold and interrupt functions will be discussed later.

The 8080 CPU does not directly indicate its internal state by transmitting a "state control" output during each state; instead, the 8080 supplies direct control output (INTE, HLDA, DBIN, $\overline{WR}$ and WAIT) for use by external circuitry.

Recall that the 8080 passes through at least three states in every machine cycle, with each state defined by successive low-to-high transitions of the $\phi 1$ clock. Pictorial 7-5 shows the timing relationships in a typical fetch machine cycle. Events that occur in each state are referenced to transitions of the $\phi 1$ and $\phi 2$ clock pulses.

The sync signal identifies the first state (T_1) in every machine cycle. As shown in Pictorial 7-5, the sync signal is related to the leading edge of the $\phi 2$ clock. There is a delay (t_{DC}) between the low-to-high transition of $\phi 2$ and the positive-going edge of the sync pulse. There also is a corresponding delay (also t_{DC}) between the next $\phi 2$ pulse and the falling edge of the sync signal. Status information is displayed on D_0-D_7 during the same $\phi 2$ to $\phi 2$ interval. Switching of the status signals is likewise controlled by $\phi 2$.

The rising edge of $\phi 2$ during T_1 also loads the processor's address lines (A_0-A_{15}). These lines become stable within a brief delay (t_{DA}) of the $\phi 2$ clocking pulse, and remain stab e until the first $\phi 2$ pulse after state T_3. This gives the processor ample time to read the data returned from memory.

Once the processor has sent an address to memory, there is an opportunity for the memory to request a wait. This is done by pulling the processor's ready line low, prior to the "ready set-up" interval (t_{RS}) which occurs during the $\phi 2$ pulse within state T_2 or T_W. As long as the ready line remains low, the processor will idle, giving the memory time to respond to the addressed data request. Refer to Pictorial 7-5.

The processor responds to a wait request by entering an alternative state (T_W) at the end of T_2, rather than proceeding directly to the T_3 state. Entry into the T_W state is indicated by a wait signal from the processor, acknowledging the memory's request. A low-to-high transition on the wait line is triggered by the rising edge of the $\phi 1$ clock and occurs within a brief delay (t_{DC}) of the actual entry into the T_W state.

A wait period may be of indefinite duration. The processor remains in the waiting condition until its ready line again goes high. A ready indication **must** precede the falling edge of the $\phi 2$ clock by a specified interval (t_{RS}), in order to guarantee an exit from the T_W state. The cycle may then proceed, beginning with the rising edge of the next $\phi 1$ clock. A WAIT interval will therefore consist of an integral number of T_W states and will always be a multiple of the clock period.

Instructions for the 8080 require from one to five machine cycles for complete execution. The 8080 sends out eight bits of status information on the data bus at the beginning of each machine cycle (during sync time). The following table defines the status information.

STATUS INFORMATION DEFINITION

Symbols	Data Bus Bit	Definition
INTA*	D_0	Acknowledge signal for interrupt request. Signal is used to gate a restart instruction onto the data bus when DBIN is active.
$\overline{WO}$	D_1	Indicates that the operation in the current machine cycle is a write memory or output function ($\overline{WO} = 0$). Otherwise, a ready memory or input operation will be executed.

STACK D_2 Indicates that the address bus holds the pushdown stack address from the Stack Pointer.

HLTA D_3 Acknowledge signal for halt instruction.

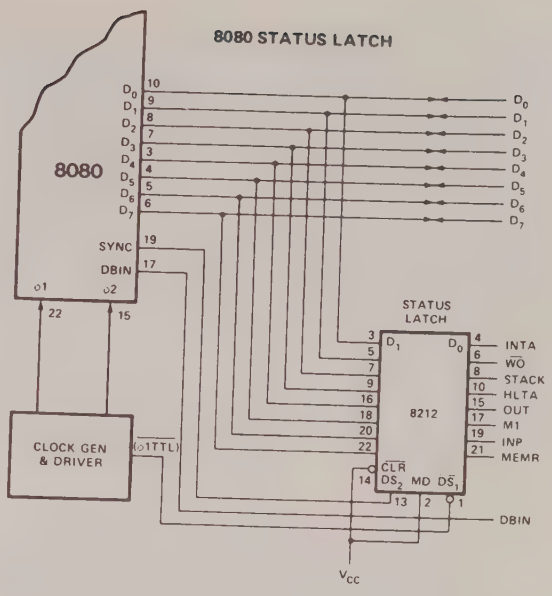
OUT D_4 Indicates that the address bus contains the address of an output device and the data bus contains the output data when $\overline{WR}$ is active.

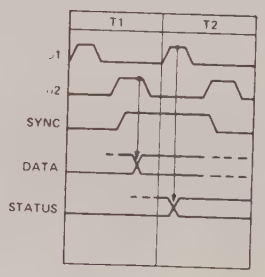
M_1 D_5 Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instruction.

INP* D_6 Indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when DBIN is active.

MEMR* D_7 Designates that the data bus is used for memory read data.

*These three status bits can be used to control the flow of data onto the 8080 data bus.





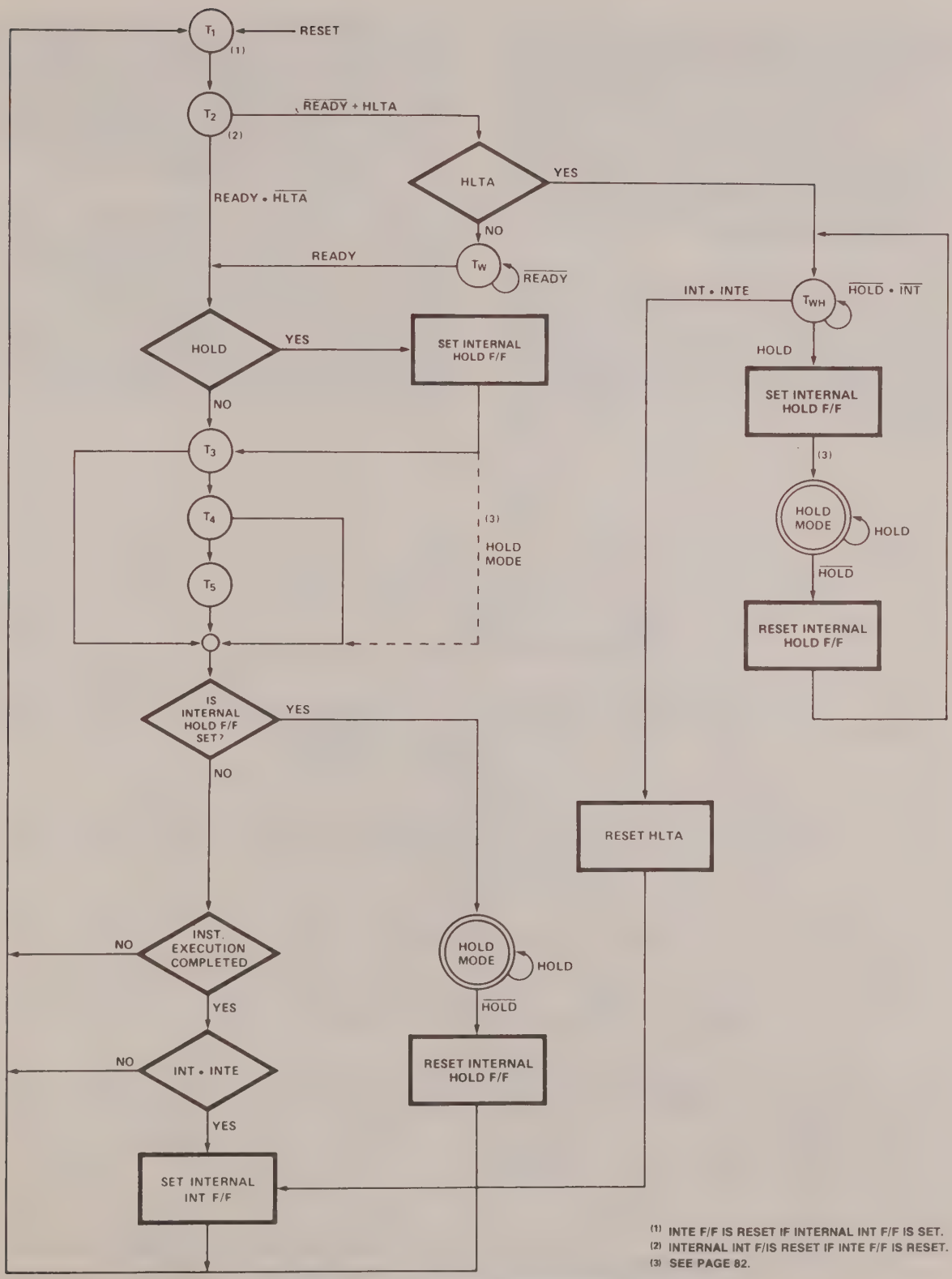
STATUS WORD CHART

TYPE OF MACHINE CYCLE

DATA BUS BIT	STATUS INFORMATION	① INSTRUCTION FETCH	② MEMORY READ	③ MEMORY WRITE	④ STACK READ	⑤ STACK WRITE	⑥ INPUT READ	⑦ OUTPUT WRITE	⑧ INTERRUPT ACKNOWLEDGE	⑨ HALT ACKNOWLEDGE	⑩ INTERRUPT ACKNOWLEDGE WHILE HALT
D_0	INTA	0	0	0	0	0	0	0	1	0	1
D_1	$\overline{WO}$	1	1	0	1	0	1	0	1	1	1
D_2	STACK	0	0	0	1	1	0	0	0	0	0
D_3	HLTA	0	0	0	0	0	0	0	0	1	1
D_4	OUT	0	0	0	0	0	0	1	0	0	0
D_5	M_1	1	0	0	0	0	0	0	1	0	1
D_6	INP	0	0	0	0	0	1	0	0	0	0
D_7	MEMR	1	1	0	1	0	0	0	0	1	0

Ⓝ STATUS WORD

TABLE 7-1 8080 Status Bit Definitions



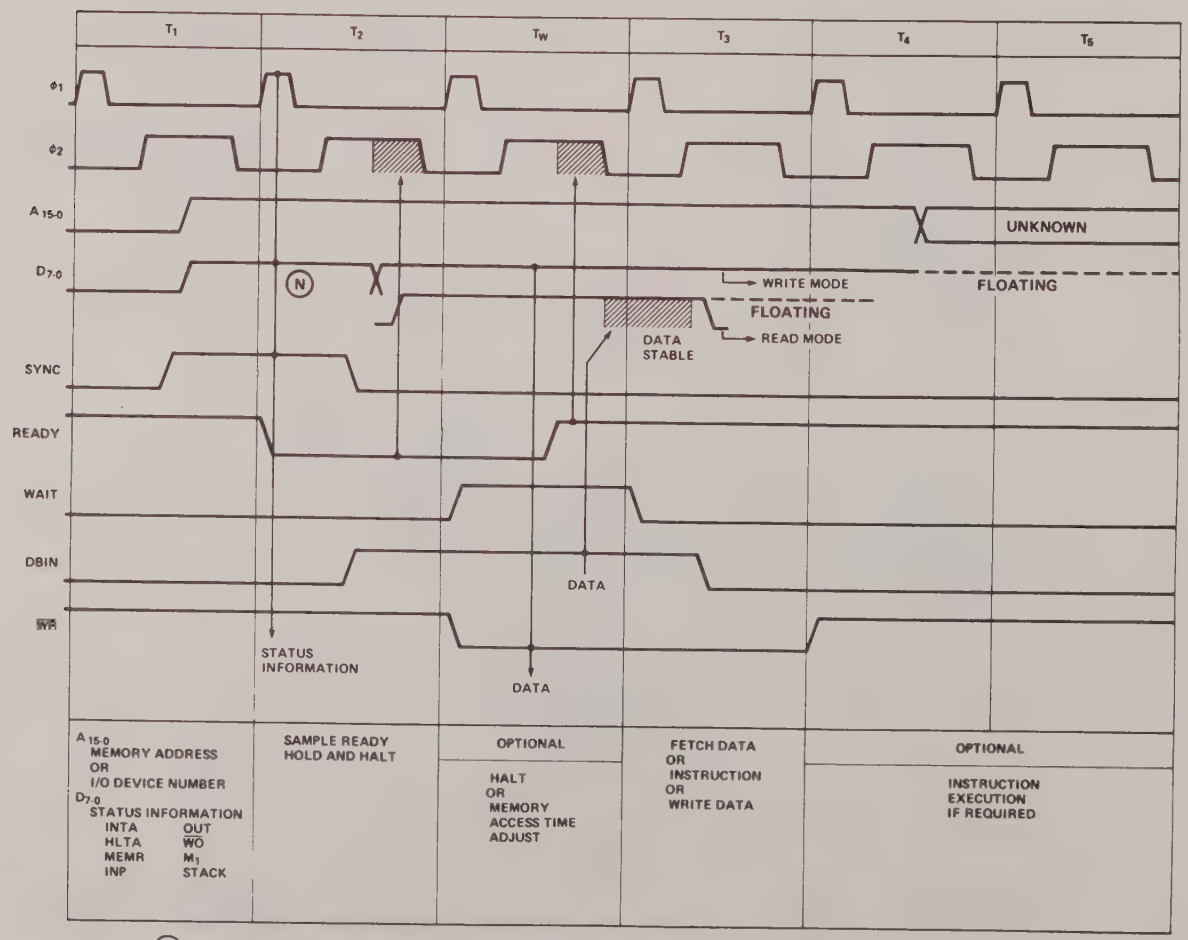
(1) INTE F/F IS RESET IF INTERNAL INT F/F IS SET.
(2) INTERNAL INT F/IS RESET IF INTE F/F IS RESET.
(3) SEE PAGE 82.

PICTORIAL 7-4 CPU State Transition Diagram

The events that take place during the T_3 state are determined by the kind of machine cycle in progress. In a fetch machine cycle, the processor interprets the data on its data bus as an instruction. During a memory read or a stack read, data on this bus is interpreted as a data word. The processor outputs data on this bus during a memory write machine cycle. During I/O operations, the processor may either transmit or receive data, depending on whether an output or an input operation is involved.

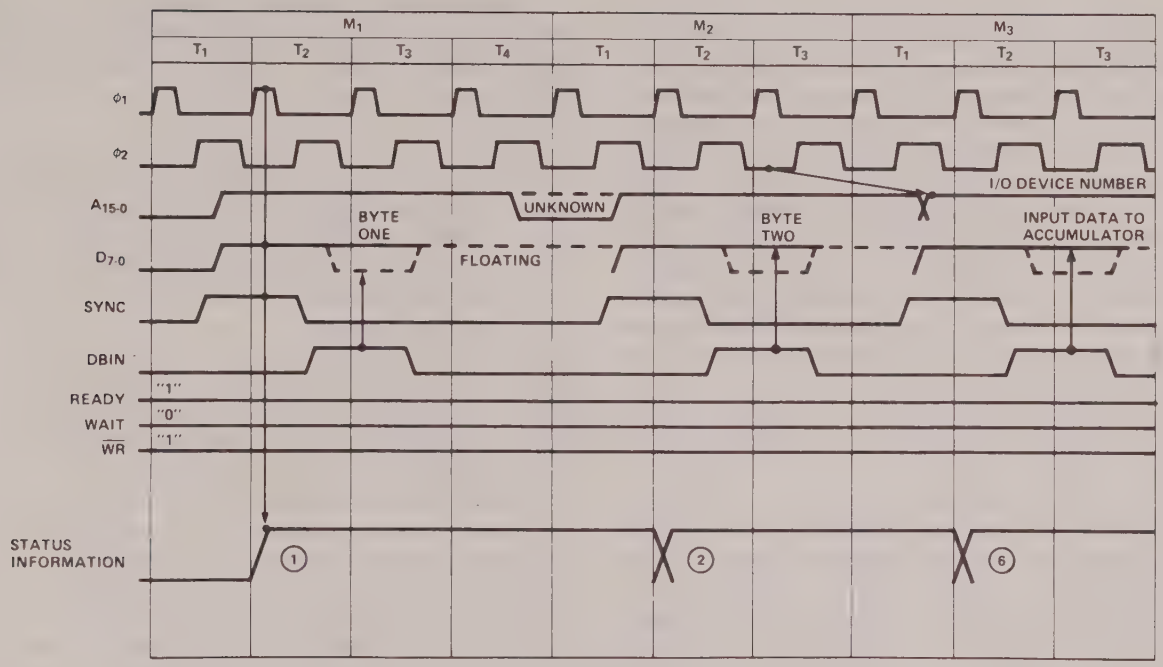
Pictorial 7-6 illustrates the timing that is characteristic of a data input operation. As shown, the low-to-high transition of $\phi 2$ during T_2 clears status information from the processor's data lines, preparing these lines for the receipt of incoming data. The data presented to the processor must have stabilized prior to both the "$\phi 1$ — data set-up" interval (t_{DS1}), that precedes the falling edge of the $\phi 1$ pulse defining state T_3, and the "$\phi 2$ — data set-up" interval (t_{DS2}), that precedes the rising edge of $\phi 2$ in state T_3. This same data must remain stable during the "data hold" interval (t_{DH}) that occurs following the rising edge of the $\phi 2$ pulse. Data placed on these lines by memory or by other external devices is sampled during T_3.

During the input of data to the processor, the 8080 generates a DBIN signal which is used externally to enable the transfer. Machine cycles in which DBIN is available include: fetch, memory read, stack read, and interrupt. DBIN is initiated by the rising edge of $\phi 2$ during state T_2 and terminated by the corresponding edge of ϕ_2 during T_3. Any T_W phases intervening between T_2 and T_3 will therefore extend DBIN by one or more clock periods.



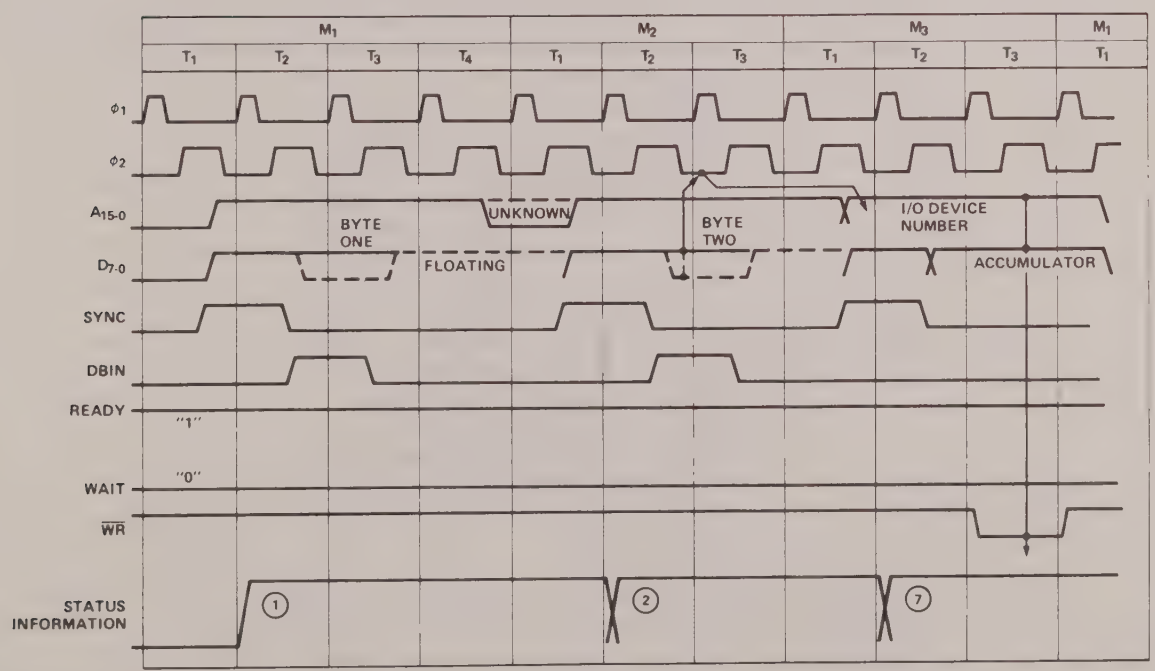
NOTE: (N) Refer to Status Word Chart on Page 73.

PICTORIAL 7-5 Basic 8080 Instruction Cycle



NOTE: (N) Refer to Status Word Chart on Page 73.

PICTORIAL 7-6 Input Instruction Cycle



NOTE: (N) Refer to Status Word Chart on Page 73.

PICTORIAL 7-7 Output Instruction Cycle

Pictorial 7-7 shows the timing of a machine cycle in which the processor outputs data. Output data is destined either for memory or for peripherals. The rising edge of $\phi 2$ within state T_2 clears status information from the CPU's data lines, and loads in the data which is to be output to external devices. This substitution takes place within the "data output delay" interval (t_{DD}) following the $\phi 2$ clock's leading edge. Data on the bus remains stable throughout the remainder of the machine cycle, until replaced by updated status information in the subsequent T_1 state. A ready signal is necessary to complete an output machine cycle. Unless such an indication is present, the processor enters the T_W state, following the T_2 state. Data on the output lines remains stable in the interim, and the processing cycle will not proceed until the ready line again goes high.

The 8080 CPU generates a $\overline{WR}$ output for the synchronization of external transfers, during those machine cycles in which the processor outputs data. These include memory write, stack write, and output. The negative-going leading edge of $\overline{WR}$ is referenced to the rising edge of the first $\phi 1$ clock pulse following T_2, and occurs within a brief delay (t_{DC}) of that event. $\overline{WR}$ remains low until re-triggered by the leading edge of $\phi 1$ during the state following T_3. Note that any T_W states intervening between T_2 and T_3 of the output machine cycle will necessarily extend $\overline{WR}$, in much the same way that DBIN is affected during data input operations.

All processor machine cycles consists of at least three states: T_1, T_2, and T_3 as just described. If the processor has to wait for a response from the peripheral or memory with which it is communicating, the machine cycle may also contain one or more T_W states. During the three basic states, data is transferred to or from the processor.

After the T_3 state, however, it becomes difficult to generalize. T_4 and T_5 states are available, if the execution of a particular instruction requires them. But not all machine cycles make use of these states. It depends upon the kind of instruction being executed, and on the particular machine cycle within the instruction cycle. The processor will terminate any machine cycle as soon as its processing activities are completed, rather than proceeding through the T_4 and T_5 states every time. Thus the 8080 may exit a machine cycle following the T_3, the T_4, or the T_5 state and proceed directly to the T_1 state of the next machine cycle.

STATE	ASSOCIATED ACTIVITIES
T_1	A memory address or I/O device number is placed on the address bus ($A_{15\times0}$); status information is placed on data bus ($D_{7\times0}$).
T_2	The CPU samples the ready and hold inputs and checks for halt instruction.
T_W (optional)	Processor enters wait state if ready is low, or if halt instruction has been executed.
T_3	An instruction byte (fetch machine cycle), data byte (memory read, stack read) or interrupt instruction (interrupt machine cycle) is input to the CPU from the DATA Bus; or a data byte (memory write, stack write or output machine cycle) is output onto the data bus.
T_4 T_5 (optional)	States T_4 and T_5 are available if the execution of a particular instruction requires them; if not, the CPU may skip one or both of them. T_4 and T_5 are only used for internal processor operations.

TABLE 7-2 State Definitions

INTERRUPT SEQUENCES

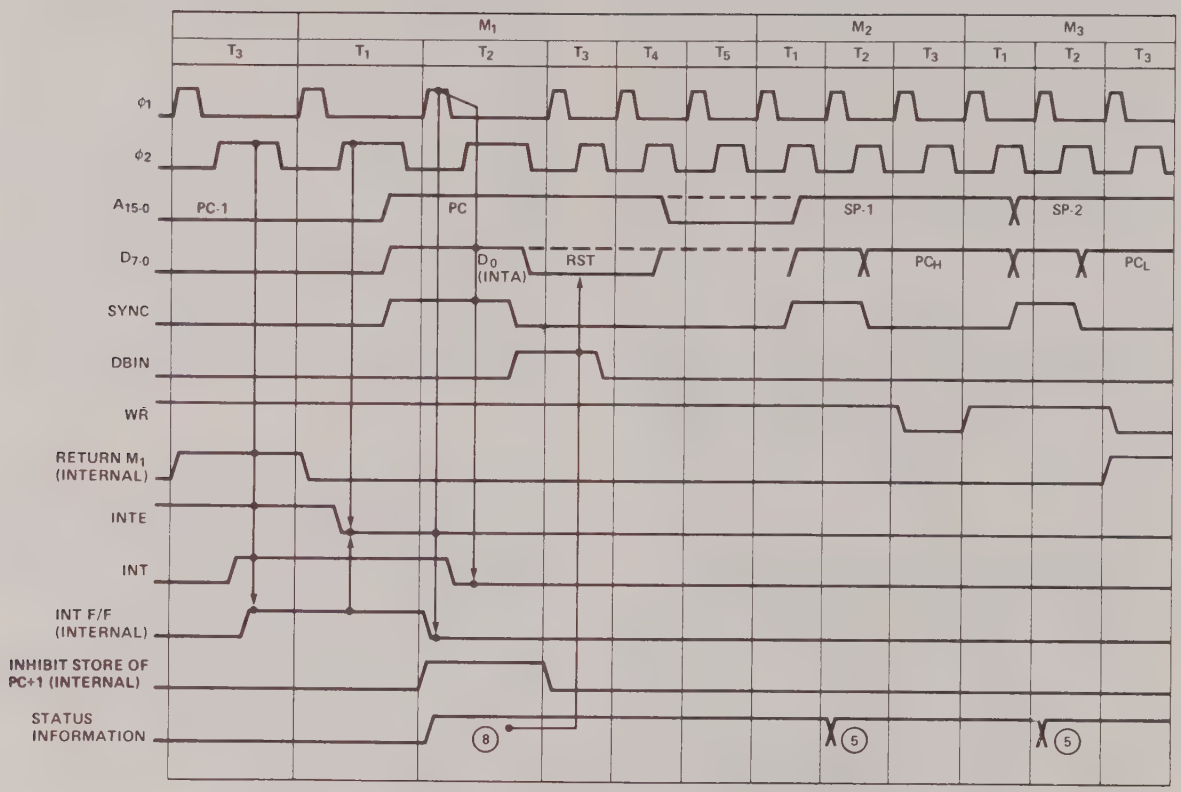
The 8080 has the built-in capacity to handle external interrupt requests. A peripheral device can initiate an interrupt by driving the processor's interrupt (INT) line high.

The interrupt (INT) input is asynchronous, and a request may therefore originate at any time during any instruction cycle. Internal logic re-clocks the external request, so that a proper correspondence with the driving clock is established. As Pictorial 7-8 (on Page 80) shows, an interrupt request (INT) arriving during the time the interrupt enable line (INTE) is high, acts in coincidence with the $\phi 2$ clock to set the internal interrupt latch. This event takes place during the last state of the instruction cycle in which the request occurs, thus ensuring that any instruction in progress is completed before the interrupt can be processed.

The interrupt machine cycle which follows the arrival of an enabled interrupt request resembles an ordinary fetch machine cycle in most respects. The M_1 status bit is transmitted as usual during the sync interval. It is accompanied, however, by an INTA status bit (D_0) which acknowledges the external request. The contents of the program counter are latched onto the CPU's address lines during T_1, but the counter itself is not incremented during the interrupt machine cycle, as it otherwise would be. In this way, the pre-interrupt status of the program counter is preserved, so data in the counter may be restored by the interrupted program after the interrupt request has been processed.

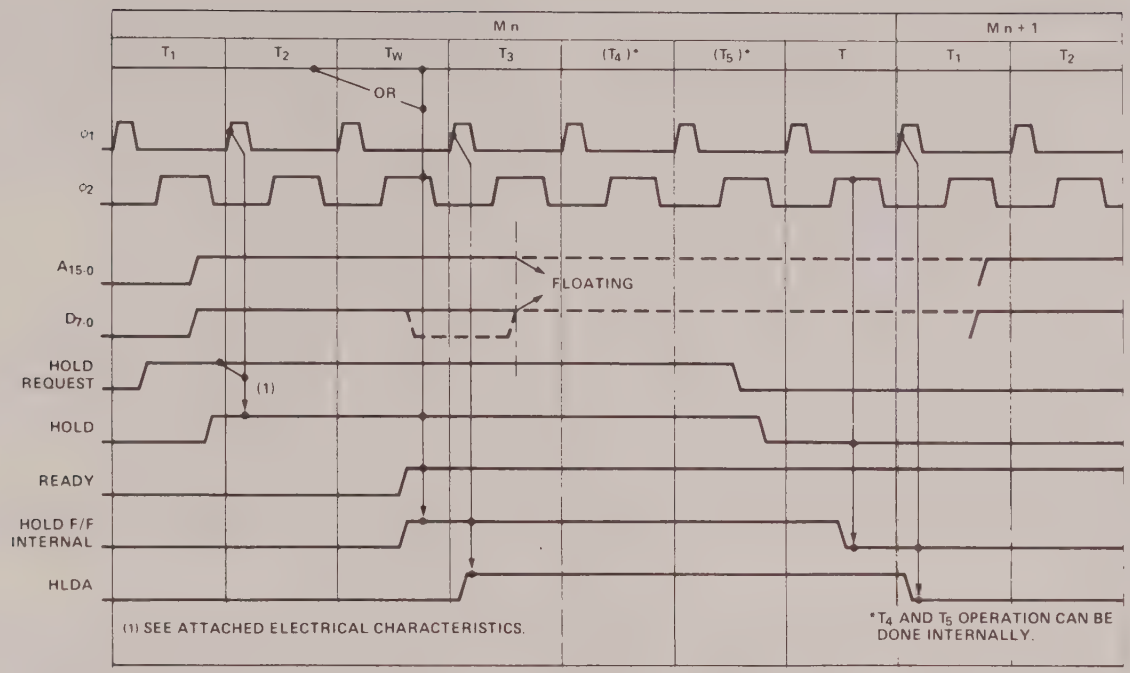
The interrupt cycle is otherwise indistinguishable from an ordinary fetch machine cycle. The processor takes no further special action. It is the responsibility of the peripheral logic to see that an 8-bit interrupt instruction is "jammed" onto the processor's data bus during state T_3. In a typical system, this means that the data-in bus from memory is temporarily disconnected from the processor's main data bus, so that the interrupting device can command the main bus without interference.

The 8080's instruction set provides a special 1-byte call which facilitates the processing of interrupts (the ordinary program call takes three bytes). This is the restart instruction (RST). A variable 3-bit field embedded in the 8-bit field of the RST enables the interrupting device to direct a call to one of eight fixed memory locations. The decimal addresses of these dedicated locations are: 0, 8, 16, 24, 32, 40, 48, and 56. Any of these addresses may be used to store the first instruction(s) of a routine designed to service the requirements of an interrupting device. Since the (RST) is a call, completion of the instruction also stores the old program counter contents on the stack.

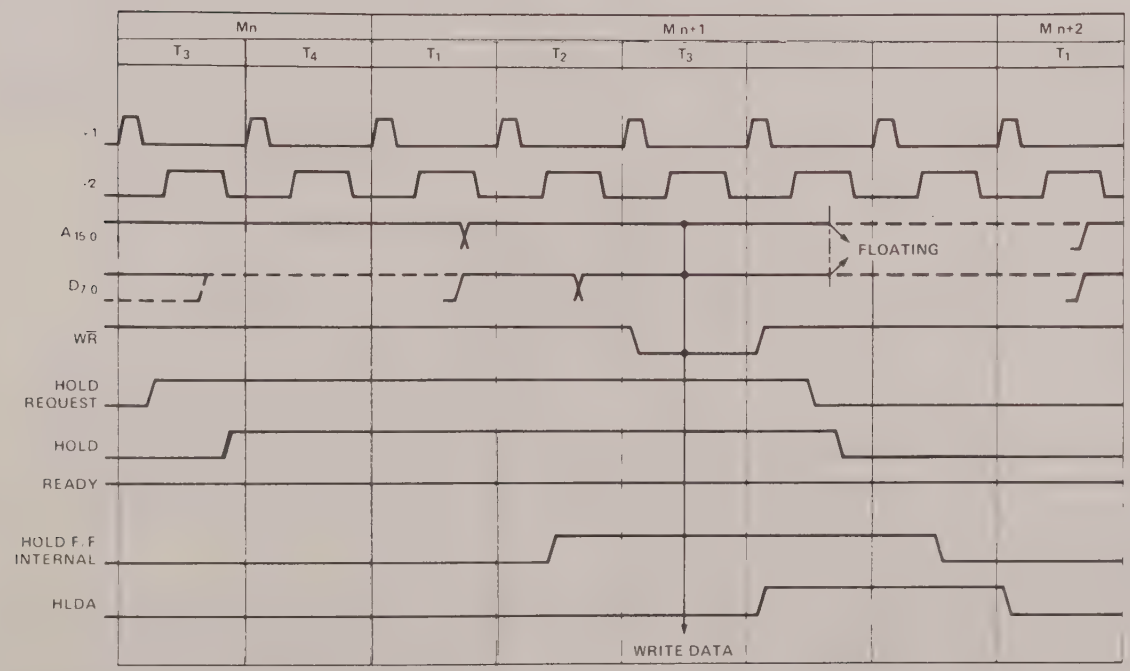


NOTE: (N) Refer to Status Word Chart on Page 73.

PICTORIAL 7-8 Interrupt Timing



PICTORIAL 7-9 HOLD Operation (Read Mode)



PICTORIAL 7-10 HOLD Operation (Write Mode)

HOLD SEQUENCES

The 8080A contains provisions for Direct Memory Access (DMA) operations. By applying a hold to the appropriate control pin on the processor, an external device can cause the CPU to suspend its normal operations and relinquish control of the address and data buses. The processor responds to a request of this kind by floating its address to other devices sharing the buses. At the same time, the processor acknowledges the hold by placing a high on its HLDA output pin. During an acknowledge hold the address and data buses are under control of the peripheral which originated the request, enabling it to conduct memory transfers without processor intervention.

Like the interrupt, the hold input is synchronized internally. A hold signal must be stable prior to the "hold set-up" interval (t_{HS}), that precedes the rising edge of $\phi 2$.

Pictorials 7-9 and 7-10 illustrate the timing involved in hold operations. Note the delay between the asynchronous hold request and the re-clocked hold. As shown in the diagram, a coincidence of the ready, hold, and $\phi 2$ clocks sets the internal hold latch. Setting the latch enables the subsequent rising edge of the $\phi 1$ clock pulse to trigger the HLDA output.

Acknowledgement of the hold request precedes slightly the actual floating of the processor's address and data lines. The processor acknowledges a hold at the beginning of T_3, if a read or an input machine cycle is in progress (see Pictorial 7-9). Otherwise, acknowledgement is deferred until the beginning of the state following T_3 (see Pictorial 7-10). In both cases, however, the HLDA goes high within a specified delay (t_{DC}) of the rising edge of the selected $\phi 1$ clock pulse. Address and data lines are floated within a brief delay after the rising edge of the next $\phi 2$ clock pulse. This relationship is also shown in the diagrams.

To all outward appearances, the processor has suspended its operations once the address and data buses are floated. Internally, however, certain functions may continue. If a hold request is acknowledged at T_3, and if the processor is in the middle of a machine cycle which requires four or more states to complete, the CPU proceeds through T_4 and T_5 before coming to a rest. Not until the end of the machine cycle is reached will processing activities cease. Internal processing is thus permitted to overlap the external DMA transfer, improving both the efficiency and the speed of the entire system.

The processor exits the holding state through a sequence similar to that by which it entered. A hold request is terminated asynchronously when the external device has completed its data transfer. The HLDA output returns to a low level following the leading edge of the next $\phi 1$ clock pulse. Normal processing resumes with the machine cycle following the last cycle that was executed.

HALT SEQUENCES

When a halt instruction (HLT) is executed, the CPU enters the halt state (T_{WH}) after state T_2 of the next machine cycle, as shown in Pictorial 7-11. There are only three ways in which the 8080 can exit the halt state:

- A high on the reset line always resets the 8080 to state T_1; reset also clears the program counter.
- A hold input causes the 8080 to enter the hold state, as previously described. When the hold line goes low, the 8080 re-enters the halt state on the rising edge of the next $\phi 1$ clock pulse.
- An interrupt (for example, INT goes high while INTE is enabled) causes the 8080 to exit the Halt state and enter state T_1 on the rising edge of the next $\phi 1$ clock pulse. NOTE: The interrupt enable (INTE) flag **must** be set when the halt state is entered; otherwise, the 8080 will only be able to exit via a reset signal.

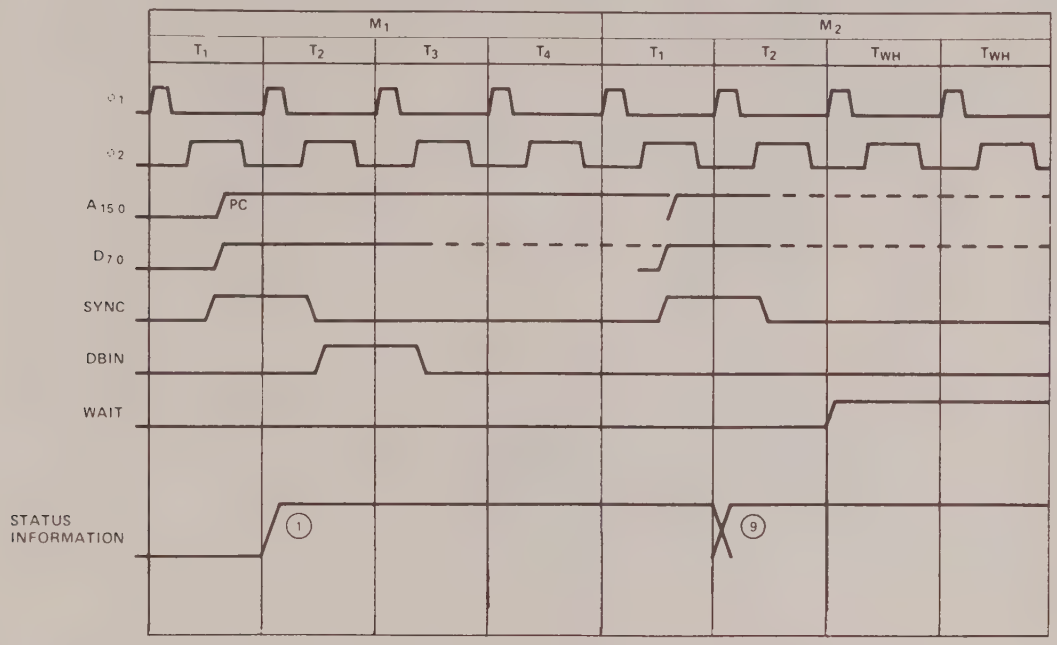
Pictorial 7-12 illustrates halt sequencing in flow chart form.

START-UP OF THE 8080 CPU

When the power is applied initially to the 8080, the processor begins operating immediately. The contents of its program counter, stack pointer, and the other working registers are naturally subject to random factors and cannot be specified. For this reason, it will be necessary to begin the power-up sequence with RESET.

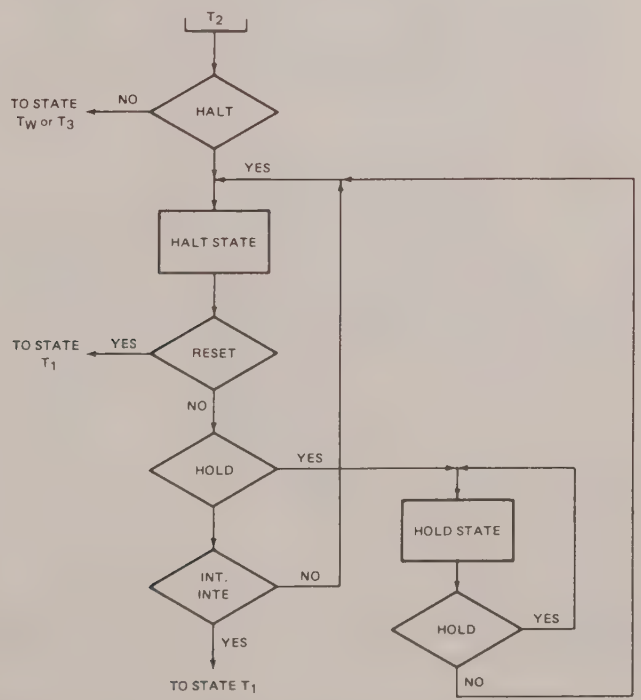
An external reset signal of three clock period duration (minimum) restores the processor's internal program counter to zero. Program execution thus begins with memory location zero, following a reset. Systems which require the processor to wait for an explicit start-up signal will store a halt instruction (EI, HLT)

in the first two locations. A manual or an automatic interrupt is used for starting. In other systems, the processor may begin executing its stored program immediately. Note, however, that the reset has no effect on status flags, or on any of the processor's working registers (accumulator, registers, or stack pointer). The contents of these registers remain indeterminate, until initialized explicity by the program.

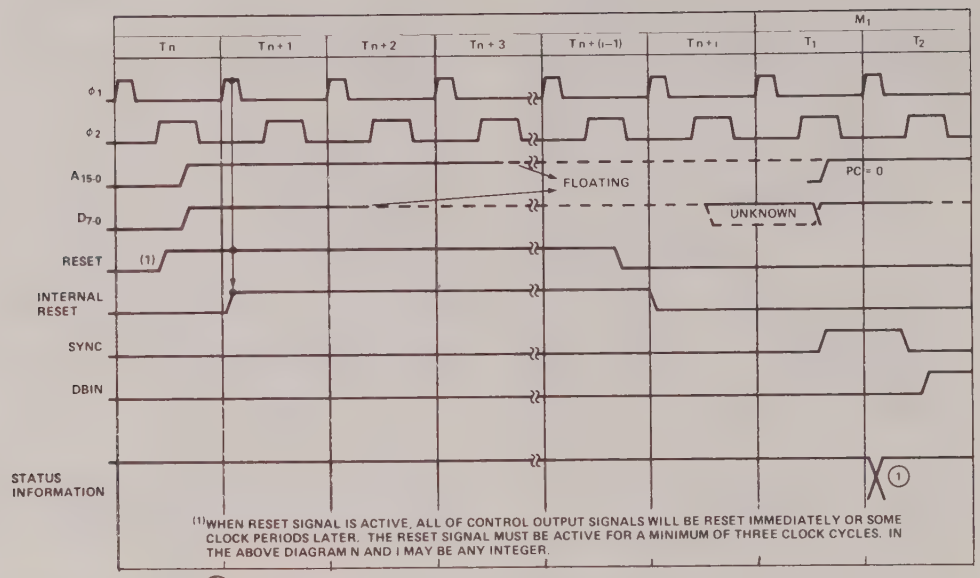


NOTE: (N) Refer to Status Word Chart on Page 73.

PICTORIAL 7-11 Halt Timing

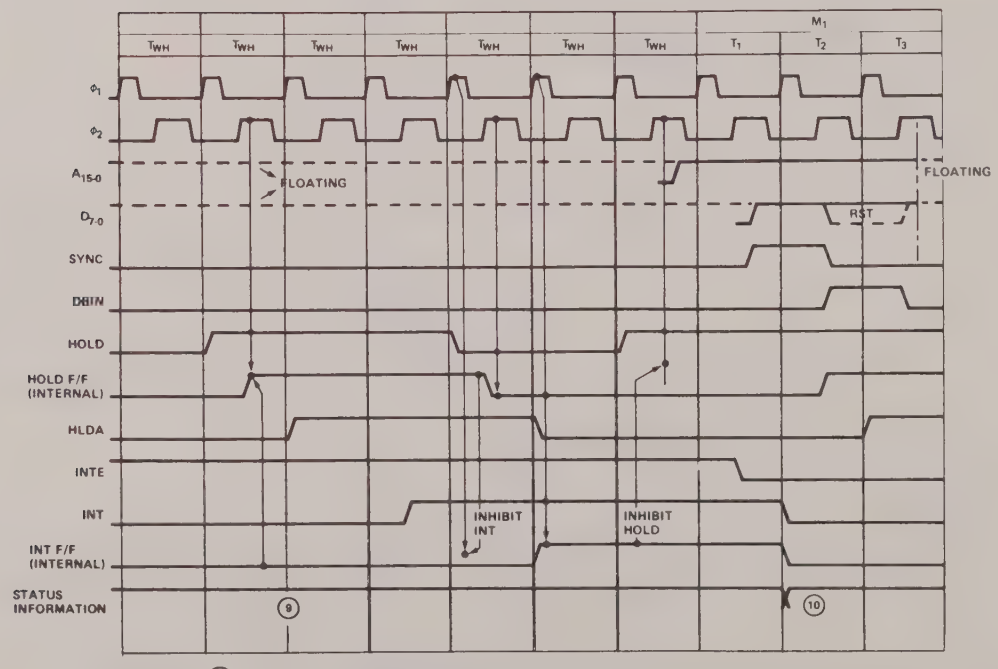


PICTORIAL 7-12 Halt Sequence Flow Chart.



NOTE: (N) Refer to Status Word Chart on Page 73.

PICTORIAL 7-13 Reset.



NOTE: (N) Refer to Status Word Chart on Page 73.

PICTORIAL 7-14 Relation between Hold and INT in the Halt State.

MNEMONIC	OP CODE		M1[1]					M2		
	$D_7 D_6 D_5 D_4$	$D_3 D_2 D_1 D_0$	T1	T2[2]	T3	T4	T5	T1	T2[2]	T3
MOV r1, r2	0 1 D D	D S S S	PC OUT STATUS	PC = PC +1	INST→TMP/IR	(SSS)→TMP	(TMP)→DDD			
MOV r, M	0 1 D D	D 1 1 0	↑	↑	↑	X[3]		HL OUT STATUS[6]	DATA →	DDD
MOV M, r	0 1 1 1	0 S S S				(SSS)→TMP		HL OUT STATUS[7]	(TMP) →	DATA BUS
SPHL	1 1 1 1	1 0 0 1				(HL) →	SP			
MVI r, data	0 0 D D	D 1 1 0				X		PC OUT STATUS[6]	B2 →	DDDD
MVI M, data	0 0 1 1	0 1 1 0				X		↑	B2 →	TMP
LXI rp, data	0 0 R P	0 0 0 1				X			PC = PC + 1 B2 →	r1
LDA addr	0 0 1 1	1 0 1 0				X			PC = PC + 1 B2 →	Z
STA addr	0 0 1 1	0 0 1 0				X			PC = PC + 1 B2 →	Z
LHLD addr	0 0 1 0	1 0 1 0				X			PC = PC + 1 B2 →	Z
SHLD addr	0 0 1 0	0 0 1 0				X		PC OUT STATUS[6]	PC = PC + 1 B2 →	Z
LDAX rp[4]	0 0 R P	1 0 1 0				X		rp OUT STATUS[6]	DATA →	A
STAX rp[4]	0 0 R P	0 0 1 0				X		rp OUT STATUS[7]	(A) →	DATA BUS
XCHG	1 1 1 0	1 0 1 1				(HL)←→(DE)				
ADD r	1 0 0 0	0 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)+(TMP)→A	
ADD M	1 0 0 0	0 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA →	TMP
ADI data	1 1 0 0	0 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →	TMP
ADC r	1 0 0 0	1 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)+(TMP)+CY→A	
ADC M	1 0 0 0	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA →	TMP
ACI data	1 1 0 0	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →	TMP
SUB r	1 0 0 1	0 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)-(TMP)→A	
SUB M	1 0 0 1	0 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA →	TMP
SUI data	1 1 0 1	0 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →	TMP
SBB r	1 0 0 1	1 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)-(TMP)-CY→A	
SBB M	1 0 0 1	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA →	TMP
SBI data	1 1 0 1	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →	TMP
INR r	0 0 D D	D 1 0 0				(DDD)→TMP (TMP) + 1→ALU	ALU→DDD			
INR M	0 0 1 1	0 1 0 0				X		HL OUT STATUS[6]	DATA → (TMP)+1 →	TMP ALU
DCR r	0 0 D D	D 1 0 1				(DDD)→TMP (TMP)+1→ALU	ALU→DDD			
DCR M	0 0 1 1	0 1 0 1				X		HL OUT STATUS[6]	DATA → (TMP)-1 →	TMP ALU
INX rp	0 0 R P	0 0 1 1				(RP) + 1 →	RP			
DCX rp	0 0 R P	1 0 1 1				(RP) - 1 →	RP			
DAD rp[8]	0 0 R P	1 0 0 1				X		(ri)→ACT	(L)→TMP, (ACT)+(TMP)→ALU	ALU→L, CY
DAA	0 0 1 0	0 1 1 1				DAA→A, FLAGS[10]				
ANA r	1 0 1 0	0 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)+(TMP)→A	
ANA M	1 0 1 0	0 1 1 0	PC OUT STATUS	PC = PC + 1	INST→TMP/IR	(A)→ACT		HL OUT STATUS[6]	DATA →	TMP

M3			M4			M5				
T1	T2[2]	T3	T1	T2[2]	T3	T1	T2[2]	T3	T4	T5
HL OUT STATUS[7]		(TMP) → DATA BUS								
PC OUT STATUS[6]	PC = PC + 1	B3 → rh								
	PC = PC + 1	B3 → W	WZ OUT STATUS[6]	DATA → A						
	PC = PC + 1	B3 → W	WZ OUT STATUS[7]	(A) → DATA BUS						
	PC = PC + 1	B3 → W	WZ OUT STATUS[6]	DATA → L WZ = WZ + 1		WZ OUT STATUS[6]	DATA → H			
PC OUT STATUS[6]	PC = PC + 1	B3 → W	WZ OUT STATUS[7]	(L) → DATA BUS WZ = WZ + 1		WZ OUT STATUS[7]	(H) → DATA BUS			
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)+CY→A									
[9]	(ACT)+(TMP)+CY→A									
[9]	(ACT)−(TMP)→A									
[9]	(ACT)−(TMP)→A									
[9]	(ACT)−(TMP)−CY→A									
[9]	(ACT)−(TMP)−CY→A									
HL OUT STATUS[7]		ALU → DATA BUS								
HL OUT STATUS[7]		ALU → DATA BUS								
(rh)→ACT	(H)→TMP (ACT)+(TMP)+CY→ALU	ALU→H, CY								
[9]	(ACT)+(TMP)→A									

MNEMONIC	OP CODE		M1[1]					M2		
	$D_7 D_6 D_5 D_4$	$D_3 D_2 D_1 D_0$	T1	T2[2]	T3	T4	T5	T1	T2[2]	T3
ANI data	1 1 1 0	0 1 1 0	PC OUT STATUS	PC = PC + 1	INST→TMP/IR	(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →TMP	
XRA r	1 0 1 0	1 S S S	↑	↑	↑	(A)→ACT (SSS)→TMP		[9]	(ACT)+(TPM)→A	
XRA M	1 0 1 0	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA →TMP	
XRI data	1 1 1 0	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →TMP	
ORA r	1 0 1 1	0 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)+(TMP)→A	
ORA M	1 0 1 1	0 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA →TMP	
ORI data	1 1 1 1	0 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →TMP	
CMP r	1 0 1 1	1 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)-(TMP), FLAGS	
CMP M	1 0 1 1	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA →TMP	
CPI data	1 1 1 1	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →TMP	
RLC	0 0 0 0	0 1 1 1				(A)→ALU ROTATE		[9]	ALU→A, CY	
RRC	0 0 0 0	1 1 1 1				(A)→ALU ROTATE		[9]	ALU→A, CY	
RAL	0 0 0 1	0 1 1 1				(A), CY→ALU ROTATE		[9]	ALU→A, CY	
RAR	0 0 0 1	1 1 1 1				(A), CY→ALU ROTATE		[9]	ALU→A, CY	
CMA	0 0 1 0	1 1 1 1				$(\bar{A})$ →A				
CMC	0 0 1 1	1 1 1 1				$\overline{CY}$ →CY				
STC	0 0 1 1	0 1 1 1				1→CY				
JMP addr	1 1 0 0	0 0 1 1				X		PC OUT STATUS[6]	PC = PC + 1 B2 →Z	
J cond addr[17]	1 1 C C	C 0 1 0				JUDGE CONDITION		PC OUT STATUS[6]	PC = PC + 1 B2 →Z	
CALL addr	1 1 0 0	1 1 0 1				SP = SP - 1		PC OUT STATUS[6]	PC = PC + 1 B2 →Z	
C cond addr[17]	1 1 C C	C 1 0 0				JUDGE CONDITION IF TRUE, SP = SP - 1		PC OUT STATUS[6]	PC = PC + 1 B2 →Z	
RET	1 1 0 0	1 0 0 1			↓	X		SP OUT STATUS[15]	SP = SP + 1 DATA →Z	
R cond addr[17]	1 1 C C	C 0 0 0			INST→TMP/IR	JUDGE CONDITION[14]		SP OUT STATUS[15]	SP = SP + 1 DATA →Z	
RST n	1 1 N N	N 1 1 1			ϕ→W INST→TMP/IR	SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1 (PCH) →DATA BUS	
PCHL	1 1 1 0	1 0 0 1			INST→TMP/IR	(HL) → PC				
PUSH rp	1 1 R P	0 1 0 1			↑	SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1 (rh) →DATA BUS	
PUSH PSW	1 1 1 1	0 1 0 1				SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1 (A) →DATA BUS	
POP rp	1 1 R P	0 0 0 1				X		SP OUT STATUS[15]	SP = SP + 1 DATA →r1	
POP PSW	1 1 1 1	0 0 0 1				X		SP OUT STATUS[15]	SP = SP + 1 DATA →FLAGS	
XTHL	1 1 1 0	0 0 1 1				X		SP OUT STATUS[15]	SP = SP + 1 DATA →Z	
IN port	1 1 0 1	1 0 1 1				X		PC OUT STATUS[6]	PC = PC + 1 B2 →Z, W	
OUT port	1 1 0 1	0 0 1 1				X		PC OUT STATUS[6]	PC = PC + 1 B2 →Z, W	
EI	1 1 1 1	1 0 1 1				SET INTE F/F				
DI	1 1 1 1	0 0 1 1				RESET INTE F/F				
HLT	0 1 1 1	0 1 1 0	↓	↓	↓	X		PC OUT STATUS	HALT MODE[20]	
NOP	0 0 0 0	0 0 0 0	PC OUT STATUS	PC = PC + 1	INST→TMP/IR	X				

M3			M4			M5				
T1	T2[2]	T3	T1	T2[2]	T3	T1	T2[2]	T3	T4	T5
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)-(TMP); FLAGS									
[9]	(ACT)-(TMP); FLAGS									
PC OUT STATUS[6]	PC = PC + 1 B3 →W									WZ OUT STATUS[11] (WZ) + 1 → P
PC OUT STATUS[6]	PC = PC + 1 B3 →W									WZ OUT STATUS[11,12] (WZ) + 1 → P
PC OUT STATUS[6]	PC = PC + 1 B3 →W		SP OUT STATUS[16]	(PCH) → DATA BUS SP = SP - 1		SP OUT STATUS[16]	(PCL) → DATA BUS			WZ OUT STATUS[11] (WZ) + 1 → P
PC OUT STATUS[6]	PC = PC + 1 B3 →W[13]		SP OUT STATUS[16]	(PCH) → DATA BUS SP = SP - 1		SP OUT STATUS[16]	(PCL) → DATA BUS			WZ OUT STATUS[11,12] (WZ) + 1 → P
SP OUT STATUS[15]	SP = SP + 1 DATA →W									WZ OUT STATUS[11] (WZ) + 1 → P
SP OUT STATUS[15]	SP = SP + 1 DATA →W									WZ OUT STATUS[11,12] (WZ) + 1 → P
SP OUT STATUS[16]	(TMP = 00NNN000) →Z (PCL) →DATA BUS									WZ OUT STATUS[11] (WZ) + 1 → P
SP OUT STATUS[16]	(rl) →DATA BUS									
SP OUT STATUS[16]	FLAGS →DATA BUS									
SP OUT STATUS[15]	SP = SP + 1 DATA →rh									
SP OUT STATUS[15]	SP = SP + 1 DATA →A									
SP OUT STATUS[15]	DATA →W		SP OUT STATUS[16]	(H) → DATA BUS		SP OUT STATUS[16]	(L) → DATA BUS	(WZ) →HL		
WZ OUT STATUS[18]	DATA →A									
WZ OUT STATUS[18]	(A) →DATA BUS									

NOTES:

1. The first memory cycle (M_1) is always an instruction fetch; the first (or only) byte, containing the op code, is fetched during this cycle.

2. If the ready input from memory is not high during T_2 of each memory cycle, the processor will enter a wait state (TW) until ready is sampled as high.

3. States T_4 and T_5 are present, as required, for operations which are completely internal to the CPU. The contents of the internal bus during T_4 and T_5 are available at the data bus; this is designed for testing purposes only. An "X" denotes that the state is present, but is only used for such internal operations as instruction decoding.

4. Only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

5. These states are skipped.

6. Memory read subcycles; an instruction or data word will be read.

7. Memory write subcycle.

8. The ready signal is not required during the second and third subcycles (M_2 and M_3). The hold signal is accepted during M_2 and M_3. The sync signal is not generated during M_2 and M_3. During the execution of DAD, M_2 and M_3 are required for an internal register-pair add; memory is not referenced.

9. The results of these arithmetic, logical, or rotate instructions are not moved into the accumulator (A) until state T_2 of the next instruction cycle. That is, A is loaded while the next instruction is being fetched; this overlapping of operations allows for faster processing.

10. If the value of the least significant 4-bits of the accumulator is greater than 9, **or** if the auxiliary carry bit is set, 6 is added to the accumulator. If the value of the most significant 4-bits of the accumulator is now greater than 9, **or** if the carry bit is set, 6 is added to the most significant 4-bits of the accumulator.

11. This represents the first subcycle (the instruction fetch) of the next instruction cycle.

12. If the condition was met, the contents of the register pair WZ are output on the address lines (A_{0-15}) instead of the contents of the program counter (PC).

13. If the condition was not met, subcycles M_4 and M_5 are skipped; the processor instead proceeds immediately to the instruction fetch (M_1) of the next instruction cycle.

14. If the condition was not met, subcycles M_2 and M_3 are skipped; the processor instead proceeds immediately to the instruction fetch (M_1) of the next instruction cycle.

15. Stack read subcycle.

16. Stack write subcycle.

17.

CONDITION	CCC
NZ — not zero (Z = 0)	000
Z — zero (Z = 1)	001
NC — no carry (CY = 0)	010
C — carry (CY = 1)	011
PO — parity odd (P = 0)	100
PE — parity even (P = 1)	101
P — plus (S = 0)	110
M — minus (S = 1)	111

18. I/O subcycle: the I/O port's 8-bit select code is duplicated on address lines 0-7 (A_{0-7}) and 8-15 (A_{8-15}).

19. Output subcycle.

20. The processor will remain idle in the halt state until an interrupt, a reset, or a hold is accepted. When a hold request is accepted, the CPU enters the hold mode; after the hold mode is terminated, the processor returns to the halt state. After a reset is accepted, the processor begins execution at memory location zero. After an interrupt is accepted, the processor executes the instruction forced onto the data bus (usually a restart instruction).

SSS or DDD	Value
A	111
B	000
C	001
D	010
E	011
H	100
L	101

rp	Value
B	00
D	01
H	10
SP	11

SPECIFICATIONS

Microprocessor	8080A.
Monitor	In on-board 1K × 8 ROM. Octal addresses, data input, and display from the front panel.
Monitor Functions	Memory display and alter. Register display and alter. Memory load and dump. Single instruction execution. Program execution. Port input and output.
Clock	2.048 MHz, crystal controlled.
Interrupts	Seven, priority vectored.
Power Supplies	+8 VDC at 10 amperes maximum. Regulated to +5 VDC on each circuit board. ±18VDC at 500 mA. Regulated to +12VDC and −5 VDC on the CPU circuit board.
Chassis Capacity	Seven locations on the mother circuit board for memory or I/O cards. Four memory cards maximum. One bus expansion location.
Operating Temperature	0° to 40°C.
Cooling	Convection type.
Power Requirements	120 VAC, 50/60 Hz, 150 watts. 240 VAC, 50/60 Hz, 150 watts.
Voltage Requirements	120 VAC (100-135 VAC), 50/60 Hz. 240 VAC (200-270 VAC), 50/60 Hz.
Dimensions	16″ W × 17.5″ D × 6.5″ H. (40.6 × 44.5 × 16.5 cm.)
Weight	21 lbs. (9.5 kg).

The Heath Company reserves the right to discontinue products and to change specifications at any time without incurring any obligation to incorporate new features in products previously sold.

SEMICONDUCTOR COMPONENT NUMBER INDEX

This Index, which shows the Heath Part Number of each semiconductor product, provides you with a cross reference between Circuit Component Numbers and their respective Heath Part Numbers. The Component Numbers are listed in numerical order.

DIODES

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
D1 / D2	100 1718
D3-D6	57-27
D101, D102	56 56
LED101-LED109	411-844
LED111-LED114	412-611

TRANSISTORS

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
Q101-Q109	417 801
Q101-Q118	417-235
Q119	417 875

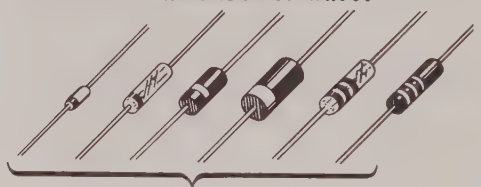
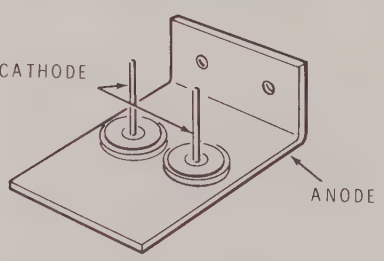
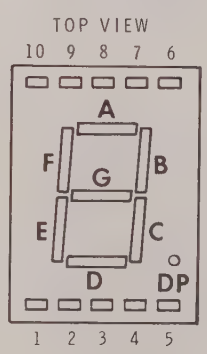
INTEGRATED CIRCUITS

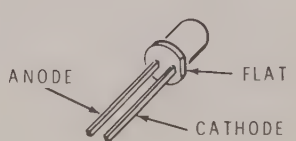
CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
IC101	442-54
IC102	443-6
IC103	443-760
IC104	443-755
IC105	443-3
IC106	443-752
IC107	443-90
IC108	443-6
IC109	443-755
IC110	443-1
IC111	443-46
IC112	443-54
IC113	443-754
IC114, IC115	443-756
IC116, IC117, IC118	443-752
IC119	443-713
IC120	442-54
IC201	442-54
IC202	442-617
IC203	442-618
IC204	444-13
IC205, IC206	443-754
IC207	443-3
IC208	443-753
IC209	443-754
IC210	443-753
IC211	443-754
IC212	443-758
IC213	443-762
IC214	443-759
IC215	443-12
IC216	443-730
IC217	443-756
IC218	443-754

SEMICONDUCTOR PART NUMBER INDEX

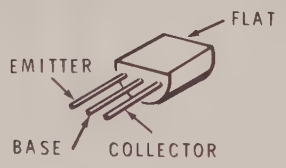
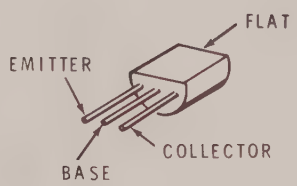
This index shows a lead configuration detail (basing diagram) for each semiconductor part number. The Heath Part Numbers are listed in numerical order.

DIODES

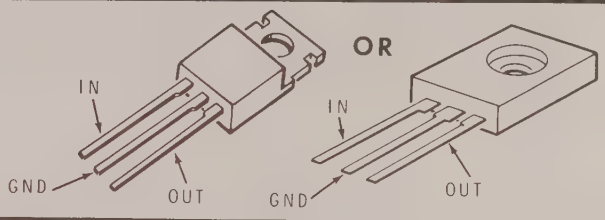
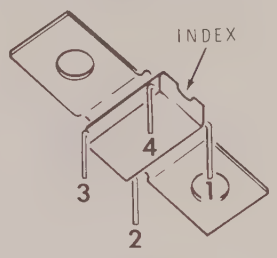
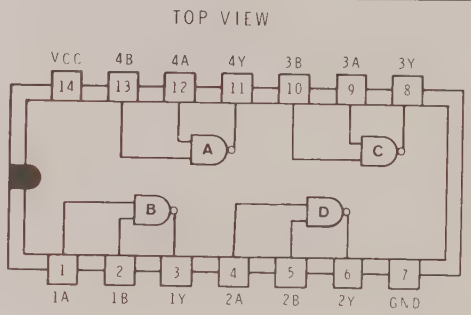
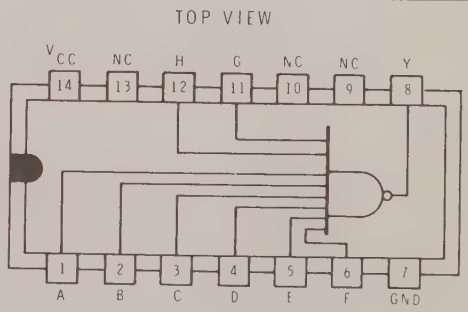
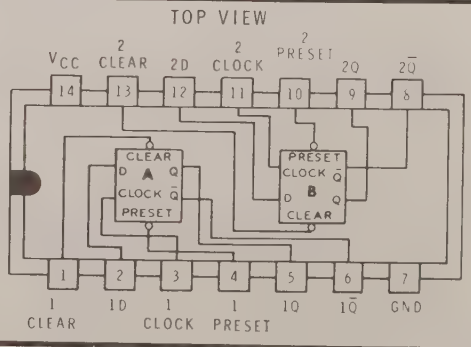
<table>
<tr><th>HEATH PART NUMBER</th><th>MAY BE REPLACED WITH</th><th>IDENTIFICATION</th></tr>
<tr><td>56-56</td><td>1N4149</td><td rowspan="2">IMPORTANT: THE BANDED END OF DIODES CAN BE MARKED IN A NUMBER OF WAYS.
 
BANDED END</td></tr>
<tr><td>57-27</td><td>1N2071</td></tr>
<tr><td>100-1708</td><td>2
HEATH PART
57-35</td><td> 
</td></tr>
<tr><td>411-844</td><td>TIL322A</td><td> 

PIN
1....SEGMENT E
2....SEGMENT D
3....COMMON CATHODE
4....SEGMENT C
5....DP
6....SEGMENT B
7....SEGMENT A
8....COMMON CATHODE
9....SEGMENT F
10....SEGMENT G</td></tr>
<tr><td>412-611</td><td>LSL-3L</td><td> 
</td></tr>
</table>

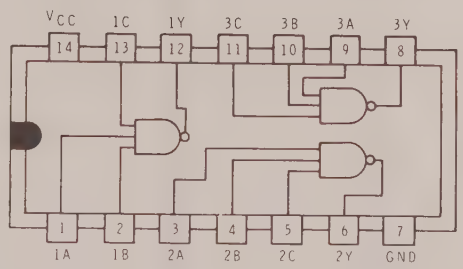
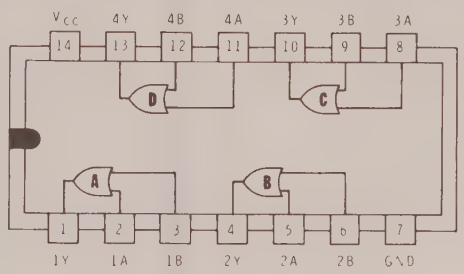
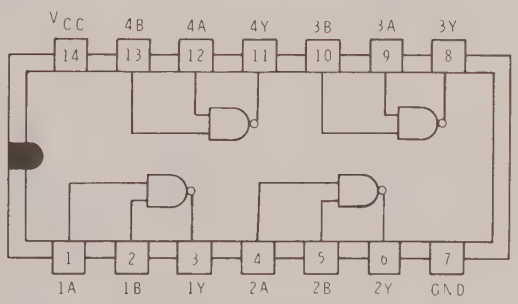
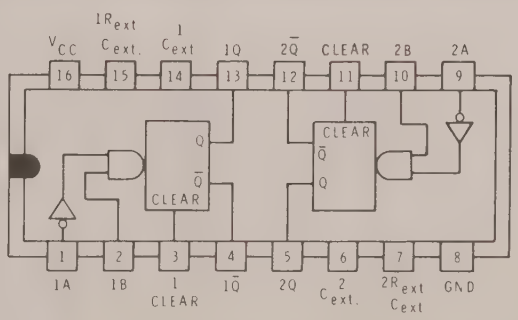
TRANSISTORS

HEATH PART NUMBER	MAY BE REPLACED WITH	IDENTIFICATION
417-235	2N4121	
417-875	2N3904	
417-881	MPSA13	

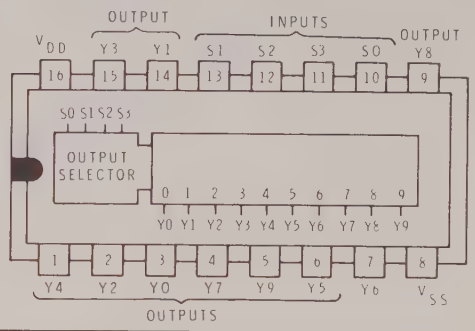
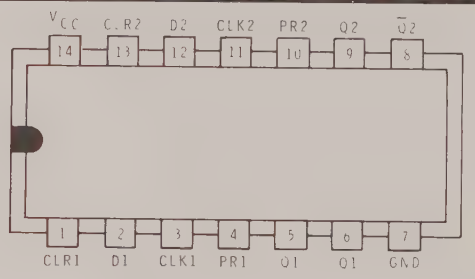
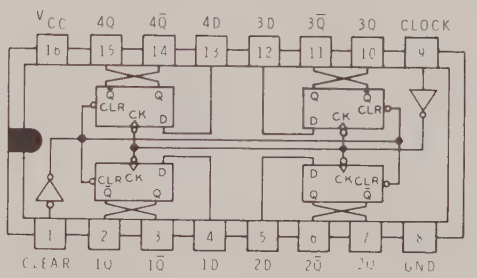
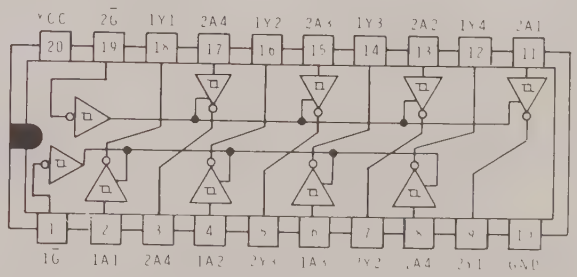
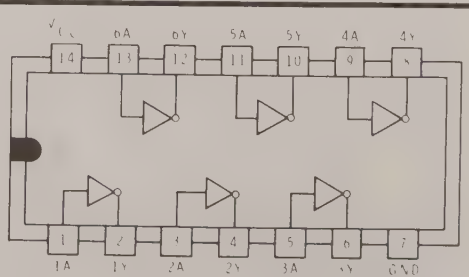
INTEGRATED CIRCUITS

HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION
442-54	7805	5 VOLT REGULATOR	
442-617	78MG	+5 TO +30 VOLT REGULATOR	
442-618	79MG	-2.2/-30 VOLT REGULATOR	
443-1	7400	QUADRUPLE 2-INPUT POSITIVE NAND GATES	
443-3	7430	8-INPUT POSITIVE NAND GATES	
443-6	7474	DUAL D-TYPE POSITIVE EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR	 TOP VIEW VCC 2 CLEAR 2D 2 CLOCK 2 PRESET 2Q 2Q̄ 14 13 12 11 10 9 8 CLEAR D A Q CLOCK Q PRESET PRESET CLOCK Q D B Q CLEAR 1 2 3 4 5 6 7 1 CLEAR 1D 1 CLOCK 1 PRESET 1Q 1Q̄ GND

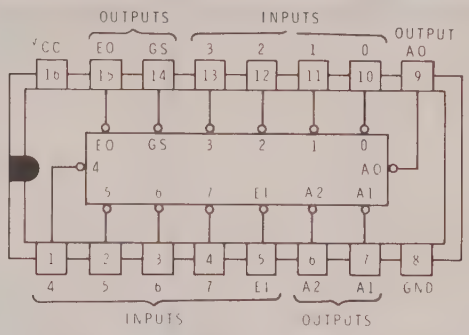
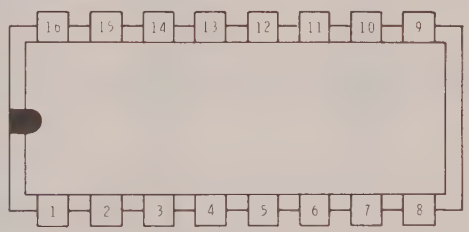
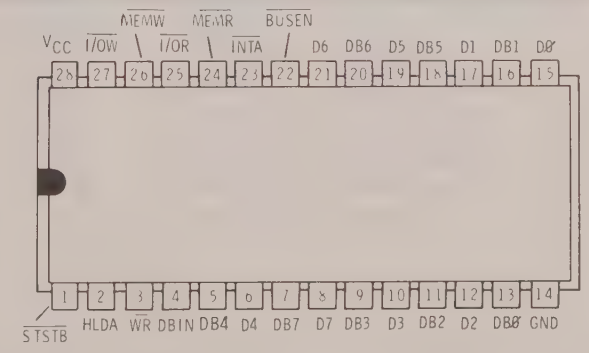
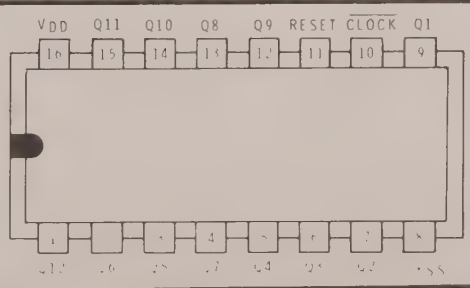
Integrated Circuits (cont'd.)

HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-12	7410	TRIPLE 3-INPUT POSITIVE-NAND GATES	
443 46	7402	QUADRUPLE 2-INPUT POSITIVE OR GATES	
443-54	7403	QUADRUBLE 2-INPUT POSITIVE NAND GATES	
443 90	74123	DUAL MONOSTABLE MULTIVIBRATOR	

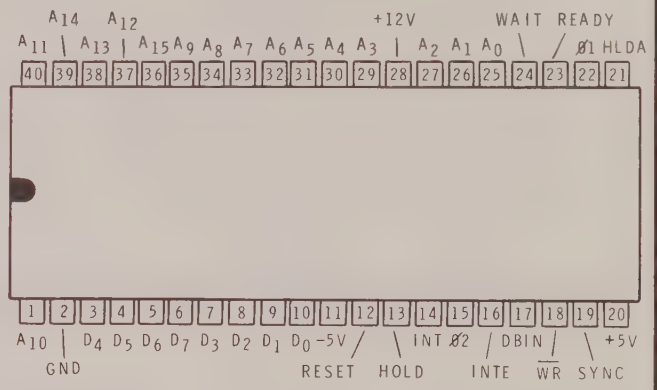
Integrated Circuits (cont'd.)

HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-713	MC14028	4-TO-10 LINE DECODER	 OUTPUT INPUTS OUTPUT V_{DD} Y3 Y1 S1 S2 S3 S0 Y8 16 15 14 13 12 11 10 9 S0 S1 S2 S3 OUTPUT SELECTOR 0 1 2 3 4 5 6 7 8 9 Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 Y9 1 2 3 4 5 6 7 8 Y4 Y2 Y0 Y7 Y9 Y5 Y6 V_{SS} OUTPUTS
443-730	74LS74	DUAL D FLIP-FLOPS	
443-752	74LS175	QUAD LATCH	
443-753	74S240	OCTAL BUFFER 3 STATE OUTPUTS	
443-754	74LS240	OCTAL BUFFERS 3 STATE OUTPUTS	
443-755	74LS04	HEX BUFFER	

Integrated Circuits (cont'd.)

HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-756	74148	8-LINE TO 3-LINE PRIORITY ENCODER	 OUTPUTS INPUTS OUTPUT VCC EO GS 3 2 1 0 A0 16 15 14 13 12 11 10 9 EO GS 3 2 1 0 4 A0 5 6 7 E1 A2 A1 1 2 3 4 5 6 7 8 4 5 6 7 E1 A2 A1 GND INPUTS OUTPUTS
443-758	8224	CLOCK GENERATOR AND DRIVER FOR 8080A CPU	 1. RESET OUTPUT 2. RESET INPUT 3. READY INPUT 4. READY OUTPUT 5. SYNC INPUT 6. O_2 CLK (TTL LEVEL) 7. STATUS STB (ACTIVE LOW) 8. GROUND (0V) 9. V_{DD} (+12V) 10. $\emptyset_2$ } 8080 CLOCK 11. $\emptyset_1$ } 8080 CLOCK 12. OSCILLATOR OUTPUT 13. TANK (USED WITH OVERTONE XTAL) 14. XTAL 2 } CONNECTIONS FOR CRYSTAL 15. XTAL 1 } CONNECTIONS FOR CRYSTAL 16. V_{CC} (+5V)
443-759	8238	SYSTEM CONTROLLER AND BUS DRIVER FOR 8080A CPU	 MEMW MEMR BUSEN VCC I/OW I/OR INTA D6 DB6 D5 DB5 D1 DB1 DB0 28 27 26 25 24 23 22 21 20 19 18 17 16 15 1 2 3 4 5 6 7 8 9 10 11 12 13 14 STSTB HLDA WR DBIN DB4 D4 DB7 D7 DB3 D3 DB2 D2 DB0 GND
443-760	MC14040	12-BIT BINARY COUNTER	

Integrated Circuits (cont'd.)

HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-762	8080A	MICRO-PROCESSOR	
444-13	MK30000	8192 BIT STATIC MOS READ ONLY MEMORY	

PARTS LIST

NOTE: The following circuit component numbers correspond to the numbers on the Schematic, X-Ray Views, and any related figures.

CHASSIS ASSEMBLY

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
C1	21-71	.001 μF (1000 pF) ceramic capacitor
C2	25-842	77,000 μF electroyltic capacitor
C3, C4	25-843	2200 μF electrolytic capacitor
D1/D2	100-1708	Diode assembly
D3-D6	57-27	1N2071 diode
F1	421-23	1-ampere, slow-blow fuse
R1	1-42-2	150 Ω, 2-watt, 10% resistor
R2	6-103	10 kΩ, 1/2-watt, 5% resistor
SP1	401-163	Speaker
SW1	60-619	Rocker switch
SW2	60-54	120/240 slide switch
SW3	60-608	NOR/LOW slide switch
T1	54-940	Power transformer
	390-1415	Instruction card
	390-1371	Keyboard label set
	432-746	25-pin plug

CONTROL CIRCUIT BOARD

CAPACITORS

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
C101	27-118	.15 μF Mylar*
C102	25-221	2.2 μF tantalum
C103	25-238	100 μF tantalum
C104	21-95	.1 μF ceramic
C105	20-107	680 pF mica
C106	25-221	2.2 μF tantalum
C107-C109	21-95	.1 μF ceramic
C110	NOT USED	
C111-C118	21-95	.1 μF ceramic

DIODE-INTEGRATED CIRCUITS (IC's)

D101	56-56	1N4149 diode
IC101	442-54	7805 IC
IC102	443-6	7474 IC
IC103	443-760	14040 IC
IC104	443-755	74LS04 IC
IC105	443-3	7430 IC
IC106	443-752	74LS175 IC
IC107	443-90	74123 IC
IC108	443-6	7474 IC
IC109	443-755	74LS04 IC
IC110	443-1	7400 IC
IC111	443-46	7402 IC
IC112	443-54	7403 IC
IC113	443-754	74LS240 IC
IC114, IC115	443-756	74148 IC
IC116-IC118	443-752	74LS175 IC
IC119	443-713	14028 IC
IC120	442-54	7805 IC

LED'S

LED101-LED109	411-844	LED display
LED110	NOT USED	
LED111-LED114	412-611	LED lamp

*DuPont Registered Trademark

TRANSISTORS

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
Q101-Q109	417-881	MPSA13
Q110	NOT USED	
Q111-Q118	417-235	2N4121
Q119	417-875	2N3904

RESISTORS

(All resistors are 1/4-watt, 5% unless otherwise listed.)

R101, R102	6-102-12	1000 Ω
R103	2-674-12	40 kΩ, 1%
R104	6-201-12	200 Ω
R105	6-561-12	560 Ω
R106, R107	6-102-12	1000 Ω
R108, R109	6-561-12	560 Ω
R110	NOT USED	
R111, R112	6-102-12	1000 Ω
R113, R114	6-561-12	560 Ω
R115, R116	6-102-12	1000 Ω
R117, R118	6-561-12	560 Ω
R119	6-102-12	1000 Ω
R120	NOT USED	
R121	6-102-12	1000 Ω
R122	6-561-12	560 Ω
R123-R129	6-150-12	15 Ω
R130	NOT USED	
R131	6-150-12	15 Ω
R132-R139	6-102-12	1000 Ω
R140	NOT USED	
R141	6-102-12	1000 Ω
R142-R149	6-561-12	560 Ω
R150	NOT USED	
R151	6-561-12	560 Ω
R152	6-102-12	1000 Ω
R153	6-273-12	27 kΩ
R154	6-151-12	150 Ω
R155	6-301-12	300 Ω
R156-R159	6-102-12	1000 Ω
R160	NOT USED	
R161-R163	6-102-12	1000 Ω
R164	6-561-12	560 Ω
R165	6-102-12	1000 Ω
R166	6-301-12	300 Ω

MISCELLANEOUS

SW101-SW116	64-839	Pushbutton switch

CPU CIRCUIT BOARD

CAPACITORS

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
C201	20-52	7.5 pF mica
C202-C204	25-221	2.2 μF tantalum
C205	21-143	.05 μF ceramic
C206, C207	25-221	2.2 μF tantalum
C208, C209	21-143	.05 μF ceramic
C210	NOT USED	
C211, C212	25-221	2.2 μF tantalum
C213-C219	21-95	.1 μF ceramic
C220	NOT USED	
C221, C222	21-95	.1 μF ceramic

DIODE-INTEGRATED CIRCUITS (IC's)

D201	56-56	1N4149 diode
IC201	442-54	7805 IC
IC202	442-617	78MGT2C IC
IC203	442-618	79MGT2C IC
IC204	444-13	MK30000 IC
IC205, IC206	443-754	74LS240 IC
IC207	443-3	7430 IC
IC208	443-753	74S240 IC
IC209	443-754	74LS240 IC
IC210	443-753	74S240 IC
IC211	443-754	74LS240 IC
IC212	443-758	8224 IC
IC213	443-762	8080A IC
IC214	443-759	8239 IC
IC215	443-12	7410 IC
IC216	443-730	74LS74 IC
IC217	443-756	74148 IC
IC218	443-754	74LS240 IC

RESISTORS

(All resistors are 1/4-watt, 5% unless otherwise listed.)

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
R201, R202	6-102-12	1000 Ω
R203	6-224-12	220 kΩ
R204-R209	6-102-12	1000 Ω
R210	NOT USED	
R211-R219	6-102-12	1000 Ω
R220	NOT USED	
R221-R229	6-102-12	1000 Ω
R230	NOT USED	
R231	2-46-12	2500 Ω, 1%
R232	2-59-12	1780 Ω, 1%
R233	2-46-12	2500 Ω, 1%
R234	2-61-12	2050 Ω, 1%
R235	6-102-12	1000 Ω

MISCELLANEOUS

Y1	404-590	18.432 MHz crystal

SERVICE INFORMATION

The following Heath Company services are available if you need them: Replacement Parts, Technical Consultation, and Factory Service. Address all correspondence to:

HEATH COMPANY
Benton Harbor, Michigan 49022

For prompt service, use a separate letter for each department you write to.

Replacement parts and repair service are also available at your nearest Authorized Service Center or Heath Electronic Center. These Centers are listed in your Heath Catalog.

REPLACEMENT PARTS

If a replacement part is needed, please include the following information in your letter:

1. Part number and description.
2. Model Number and Series Number of the equipment.

If your equipment is in the Warranty period, add:

3. Date of purchase.
4. Nature of defect.

Heath Company will fill your order promptly. Please DO NOT RETURN PARTS unless they are requested. Parts that are damaged through carelessness or misuse by the customer will not be replaced without cost.

TECHNICAL CONSULTATION

You can write to our Technical Consultants for help with any Heath equipment, or for answers to any questions about the use of this equipment.

The completeness and accuracy of the advice mailed back to you depends entirely on the information in your letter. Be sure to include:

1. The Model Number and Series Number of the equipment (on blue and white identification label).
2. Date of purchase.
3. An exact description of the difficulty. Include switch positions, connections to other units, operating procedures, voltage reading, and any other information you think might be helpful.
4. List everything you have done in attempting to correct the difficulty.

FACTORY SERVICE

If you do not have qualified repair services at your disposal, you can return your equipment to the Heath Company Service Department to have it repaired for a minimum service fee. (Equipment that has been modified will not be accepted for repair.) Refer to Shipping Instructions for details on how to package and ship the equipment.

To be eligible for replacement parts under the terms of the Warranty, equipment returned for factory service must be accompanied by the invoice or the sales slip, or a copy of either. (If you send the original invoice or sales slip, it will be returned to you.)

SHIPPING INSTRUCTIONS

Check the equipment to see that all parts are in place. Then, wrap the equipment in heavy paper. Place the equipment in a strong carton, and put at least three inches of resilient packing material (shredded paper, excelsior, etc.) on all sides between the equipment and the carton.

Seal the carton with gummed paper tape and tie it with a strong cord. Ship it by prepaid Express or insured Parcel Post to:

HEATH COMPANY
Benton Harbor, Michigan 49022

Attach a letter, containing the following information to the outside of the carton:

1. Your name and return address.
2. Date of purchase.
3. A brief description of the difficulty.
4. Your authorization to ship the repaired unit back to you C.O.D. for the service and shipping charges, plus the cost of parts not covered by the Warranty.

YOUR HEATH DATA SYSTEMS PRODUCT 90 DAY LIMITED WARRANTY

For a period of 90 days after purchase, Heath Company will repair free of charge any product that is defective either in materials or workmanship. We warrant that during the first 90 days after purchase, our products, when used in accordance with our printed instructions, will meet published specifications.

If your Heath factory-assembled product malfunctions or fails to operate at any time during the warranty period, through no fault of yours, we will service it free upon proof of purchase and delivery at your expense to the Heath factory, or any Heathkit Electronic Center (units of Schlumberger Products Corporation), or any of our authorized independent distributor service stations.

Consultation is available on any problem you might encounter in the use of your Heath product. Just drop us a line or give us a call. Sorry, we cannot accept collect calls.

Our warranty does not cover and we are not responsible for damage caused by misuse or fire or unauthorized modifications to or uses of our products for purposes other than advertised. Our warranty does not include reimbursement for customer assembly, disassembly, set-up time, or unauthorized repairs.

This warranty covers only Heath factory assembled products and is not extended to allied equipment or components used in conjunction with these products. **We are not responsible for incidental or consequential damages.** Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

If you are not satisfied with our service - warranty or otherwise, or with our products, write directly to our Director of Customer Services, Heath Company, Benton Harbor, Michigan 49022. He will make certain your problems receive immediate, personal attention.

HEATH COMPANY
Benton Harbor, Michigan 49022

The Heath Company reserves the right to discontinue products and to change specifications at any time without incurring any obligation to incorporate new features in products previously sold.

LITHO IN U.S.A.